

УДК 004.056

Шифр специальности ВАК: 2.3.6

А.В. Бочков, А.Г. Сабанов, А.А. Шелупанов

Концепция формирования метрик безопасности прикладного ПО для объектов КИИ

Актуальность. Усиление требований нормативно-правовой базы к защите информации объектов критической информационной инфраструктуры приводит к необходимости создания системы количественных оценок безопасности разрабатываемого и используемого прикладного программного обеспечения. **Цель исследования.** Разработка модернизированного метода количественных оценок безопасности прикладного программного обеспечения на основе методологии системы международных стандартов по качеству программного обеспечения. **Методы.** В работе комплексно применены методы системного анализа, формализации, синтеза, сравнения, вычислительного эксперимента и нормативной привязки. **Научная новизна.** Разработана концепция введения системы количественных оценок безопасности прикладного программного обеспечения и новой подхарактеристики безопасности программ «уровни безопасности программного обеспечения» как показателя качества разрабатываемых программ. Предложена трехуровневая система формирования уровней безопасности программного обеспечения, опирающаяся на двухконтурную модель агрегирования, в которой заградительные и градуируемые метрики играют математически различную функциональную роль. **Результаты.** Применимость предложенной концепции показана на примере расчета метрик безопасности. **Практическая значимость.** Предложенная система количественных оценок уточненных метрик безопасности может применяться в проектах создания и модернизации прикладного программного обеспечения объектов критической информационной инфраструктуры.

Ключевые слова: безопасность программного обеспечения, критическая информационная инфраструктура, SQuaRE, метрики безопасности, гармоническое среднее, заградительные метрики, модель угроз, уровни безопасности программного обеспечения.

DOI: 10.21293/1818-0442-2026-29-2-160-176

Введение

Ключевая роль в обеспечении безопасности информационных систем (ИС) различного назначения принадлежит программному обеспечению (ПО). Применительно к ПО ИС объектов *критической информационной инфраструктуры (КИИ)*, согласно ст. 11 Федерального закона от 26.07.2017 № 187-ФЗ и приказу ФСТЭК России от 25 декабря 2017 г. № 239, существенным требованием является обеспечение безопасности ПО в соответствии с основными положениями стандарта [1].

Пункт 3.1 указанного стандарта трактует термин «безопасность ПО» как «программное обеспечение, разработанное в ходе реализации совокупности процессов (мер), направленных на предотвращение появления и устранение недостатков программы». Под недостатками ПО понимается наличие ошибок, известных уязвимостей, нежелательных конструкций, недеklarированных возможностей (НДВ), недостаточной защиты от несанкционированного доступа (НСД) и скрытых каналов. При этом подразумевается, что нарушение безопасности ПО может привести к реализации угроз безопасности информации с использованием недостатков ПО, направленных на систему, для которой данное ПО разрабатывается, поскольку само по себе ПО гораздо реже является предметом непосредственной цели злоумышленников. Другими словами, в этой парадигме ПО может являться источником угроз или путем их реализации.

В целях нейтрализации связанных с ПО угроз, согласно [1], ПО должно оставаться безопасным на всем жизненном цикле (ЖЦ) – на этапах проектиро-

вания, разработки, внедрения, эксплуатации и вывода из эксплуатации [2].

В стандарте [1] степень безопасности ПО предлагается оценивать по строгости выполнения требований к организации процесса разработки, а также по полноте проведения различных видов анализа кода на всех этапах ЖЦ. Например, наличие или отсутствие уязвимостей в коде программы следует проверять на основе статического и динамического анализа, композиционного анализа, тестирования на проникновение и экспертизы кода, включая анализ самих инструментов тестирования, используемых анализаторами баз уязвимостей и настроек тестирования. Обратим внимание, что даже после выполнения перечисленного комплекса различных видов тестирования никто не подпишется, что ПО гарантированно не содержит уязвимостей, поскольку через непродолжительное время в заимствованном или собственном коде могут быть обнаружены новые уязвимости и не обнаруженные ранее другие недостатки. Поэтому в отчетах по результатам анализа кода часто присутствует сопровождающая фраза, например, «известных на данный момент уязвимостей в данном ПО не обнаружено».

Рассмотренный пример показывает, что о безопасности ПО можно судить по косвенным признакам, в данном случае – по мере соблюдения требований положений стандарта [1] на всем ЖЦ ПО, а именно в затраченных усилиях (приобретение и настройка анализаторов кода, подготовка и проведение тестирования, анализ отчетов, устранение уязвимостей, повторный анализ для появления уверенности в

отсутствии известных уязвимостей и т.д.). Применительно к безопасности ПО для объектов КИИ ФСТЭК России установил требования к процессу разработки и поиску уязвимостей на всех этапах ЖЦ, документации, порядку обновления ПО [3].

В недавно принятом методическом документе ФСТЭК России [4] операторам (владельцам) ИС рекомендуется проведение оценок защищенности систем, при этом предполагается выполнение требований стандарта [1] и проведение в основном качественного анализа безопасности ПО, в первую очередь, на отсутствие уязвимостей. Однако в практических работах все чаще возникает необходимость получения не только качественной, но и количественной оценки безопасности разрабатываемого заказного или приобретаемого ПО, особенно для объектов КИИ [5].

Одним из путей решения указанной задачи является использование подхода хорошо развитой количественной оценки качества ПО в системе стандартов ISO/IEC 25000, более известной как SQuaRE [6–12], поскольку безопасность является одной из характеристик качества ПО. Качеству ПО в указанной системе международных стандартов посвящено более 30 документов, значительная часть которых переведена на русский язык и введена в систему стандартов ГОСТа в виде идентичных (прямой перевод).

Система стандартов ISO/IEC 25000 развивается с 1991 г. по настоящее время и имеет ряд преимуществ, которые привлекают к себе для их применения для оценки качества программного обеспечения прозрачностью, измеримостью, повторяемостью и простотой оценивания метрик, позволяющих в цифровом выражении получать результат оценки качества ПО. Оценка качества ПО в указанной системе международных стандартов проводится с помощью трехуровневой системы: введены 8 основных характеристик качества (в число которых входит безопасность), каждая характеристика на втором уровне определяется рядом подхарактеристик, которые на третьем уровне имеют достаточно просто измеримые метрики, позволяющие получать их численные значения. Именно это свойство системы стандартов ISO/IEC 25000 привлекает для попытки ее использования в оценке безопасности ПО.

Для ПО систем объектов КИИ необходимость использования исключительно безопасного ПО утверждена в приказе ФСТЭК России [3]. Следовательно, для применения стандартов ISO/IEC 25000 к созданию системы количественных метрик безопасности ПО для объектов КИИ требуется коррекция подходов SQuaRE, обусловленная требованиями [3], что является целью данной работы.

Таким образом, в настоящей статье приводится обоснование введения модернизированных подхарактеристик безопасности ПО, учитывающих современное состояние теории защиты информации и определяющих причинно-следственную связь между процессами безопасной разработки ПО и характеристиками его безопасности. С использованием подхо-

дов к формированию уровней доверия [13, 14] предлагается введение уровней доверия к безопасности ПО. На основе приведенного ниже анализа научных работ и стандартов, определяющих качество ПО, разработана методика количественной оценки безопасности разрабатываемого ПО как показателя (характеристики) качества ПО. В настоящей статье рассматривается прикладное ПО, поскольку оно наиболее часто разрабатывается для объектов КИИ.

Обзор научных работ по оценке безопасности ПО и анализ недостатков SQuaRE

Отечественные исследования. Из работ отечественных авторов выделим фундаментальные исследования. Вопросы исследования безопасности ПО рассматривались в работах И.Б. Шубинского, в том числе применительно к АСУ ТП железнодорожного транспорта [15–17]. В работе [15] рассмотрены основные модели надежности функционирования ПО с использованием стандарта качества ПО [9]. Также установлено, что для объектов КИИ требования к качеству ПО определяются рисками. В частности, показано, что для критических систем чрезвычайно важно обеспечение целостности и доступности ПО.

В работе [16] показано, что комплексная безопасность ПО в АСУ ТП должна включать в себя качество и функциональную безопасность, отсутствие НДВ и защищенность от НСД. В работе [17] рассмотрены требования к информационной безопасности ПО, которые не должны содержать вредоносного кода, вирусов, НДВ, нарушений целостности ПО.

Значительный вклад в развитие теории безопасного ПО и методов оценки безопасности ПО внес коллектив исследователей под руководством А.С. Маркова. Так, в работе [18] рассмотрены математические модели полноты тестирования, сложности ПО, временные модели роста надежности и отладочные модели. К сожалению, модели оценки безопасности ПО в данной работе не рассматривались. В работе [19] сформулировано определение БПО: безопасным ПО называется ПО, разработанное с использованием совокупности мер, направленных на предотвращение появления и устранение уязвимостей программы. Также показано, что в оценках безопасности ПО много субъективного от экспертов-оценщиков.

В работе [20] отмечено, что наиболее актуальными моделями представления программ с позиции безопасности являются абстрактное синтаксическое дерево и абстрактный синтаксический граф. Методика основывается на поиске программных дефектов в программном коде путем сопоставления фрагментов кода с образцами из базы данных шаблонов (эвристических правил) дефектов безопасности. Разработана метамодель, позволяющая формализовать описание эвристик поиска дефектов на различных уровнях представления кода, что обеспечивает повышение унификации выявления сложных уязвимостей. Показано, что эвристический анализ может служить базой при использовании различных техник аудита БПО.

Оценка безопасности ПО затронута в работах [21, 22], при этом, как правило, использовались или модернизировались актуальные на время написания работ модели качества системы стандартов SQuaRE без акцента на метрики безопасности ПО.

Зарубежные исследования. За рубежом в последние годы развивается подход SSA (Software Security Assurance). Обеспечение гарантий безопасности программного обеспечения – это процесс, направленный на то, чтобы ПО работало с уровнем безопасности, соответствующим потенциальному ущербу, который может быть вызван потерей, неточностью, изменением, недоступностью или неправомерным использованием данных и ресурсов, находящихся под контролем и защитой ПО [23]. В процессе разработки ПО это выполнение требований безопасности, начиная с ранних стадий проектирования, при кодировании и внедрении ПО. Известны работы Кэрола Вуди (Carol Woody), развивающего подход SSA на основе анализа угроз безопасности, связанных с ПО [24].

Одним из продвигаемых на рынке решений SSA является защищенная платформа разработки веб-приложений Рефлективиз [25]. Тем не менее в условиях роста кибератак, согласно отчету State of Web Exposure 2026 [26], реальная защищенность веб-ресурсов снижается. Так, 64% сторонних приложений получают доступ к конфиденциальным данным через Web-доступ по сравнению с 51% в 2024 г. Между тем опрос руководителей служб безопасности выявил серьезный пробел: хотя 81% специалистов по безопасности считают веб-атаки приоритетной угрозой, только 39% из них используют специальные решения для парирования этих угроз.

Другим развивающимся направлением исследований является Dep (Defect Prediction in Software – прогнозирование недостатков ПО). Модели Dep создаются с использованием двух подходов: с использованием измеримых свойств программной системы, называемых программными метриками, и с использованием данных о недостатках ПО из аналогичного программного проекта. Изначально в Dep использовались только статические методы анализа кода, но затем для отслеживания поведения ПО на разных стадиях ЖЦ стали интенсивно применяться методы машинного обучения. Например, некоторые авторы использовали дерево решений (DT) [27] и искусственную нейронную сеть (ANN) [28] для построения моделей Dep с использованием объектно-ориентированных (ОО) метрик [29].

Стратегия Zero Trust основывается на принципе «никогда не доверяй, всегда проверяй» (never trust, always verify), при котором ни один пользователь, устройство, пакет данных или сетевой интерфейс по умолчанию не считаются надёжными, а любой элемент инфраструктуры или учётная запись рассматриваются как потенциально небезопасные и подлежат строгой верификации. Данная концепция была в 2010 г. сформулирована Д. Киндервагом, аналитиком Forrester Research [30]. Этот подход стал весьма актуален и распространен в годы эпидемии ковид-19 в

условиях массового перехода на удалённую работу, чтобы сотрудники пользовались корпоративными ресурсами с личных устройств.

Анализ модели качества SQuaRE и её недостатков

Наиболее известным и традиционным подходом является оценка ПО и безопасности как одной из характеристик модели качества согласно стандартам SQuaRE. Так, в работе [31] показано, что общепринятое определение качества программного обеспечения является не всегда адекватным и должно быть реализовано с использованием модели качества, которая уточняет характеристики в соответствии с рисками и ожиданиями заинтересованных сторон в отношении ПО в их индивидуальном контексте. Некоторые из последних результатов исследований подходов к оценке качества ПО представлены Champion et al. [32]. В этой работе выполнен обзор 75 релевантных научных статей, в котором выделено пять различных подходов, в которых качество рассматривается как эвристика, как ремонтпригодность, как целостная концепция, как структурные особенности ПО и как надёжность его функционирования. При этом, как показано в данном обзоре, система оценки характеристик качества ПО, принятая в стандартах SQuaRE, упоминается практически во всех рассмотренных работах.

Ключевые проблемы существующих метрик безопасности SQuaRE

Проведённый анализ позволяет выделить следующие фундаментальные недостатки текущей системы метрик безопасности в стандартах SQuaRE, которые однозначно вытекают из обзора:

1. *Неполнота покрытия требований российских регуляторов.* Стандарты SQuaRE не учитывают требования ГОСТ Р 56939 [1] в части обязательности проверки ПО на наличие НДВ (недекларированных возможностей) и нежелательных конструкций. Отсутствуют метрики, оценивающие наличие утверждённой модели угроз (МУ), что является обязательным требованием для объектов КИИ согласно [3].

2. *Отсутствие метрик безопасности самого кода ПО.* В стандарте [11] безопасность ПО представлена подхарактеристиками целостности, конфиденциальности, неотказуемости, отслеживаемости и аутентичности. Однако отсутствует явная подхарактеристика, оценивающая безопасность исходного и исполняемого кода: наличие уязвимостей, результатов статического (SAST) и динамического (DAST) анализа, анализа состава кода (SCA), тестов на проникновение. Как показано в работах [19, 20], именно качество кода является первичным источником угроз.

3. *Использование среднего арифметического для агрегирования.* В стандарте [11] и большинстве работ используется среднее арифметическое для свёртки метрик [21]. Как убедительно показано в работе [32], это приводит к эффекту маскировки (компенсации), когда критический провал по одной метрике (например, $K_1 = 0$ – отсутствие модели угроз) компенсируется высокими значениями других мет-

рик, давая формально приемлемую оценку. Это недопустимо для объектов КИИ.

4. *Недостаточность метрик неотказуемости и аутентификации.* В стандарте [11] для неотказуемости предусмотрена только одна метрика – «использование цифровой подписи». Однако этого недостаточно – требуются также метрики идентификации, аутентификации согласно стандартам [13, 14], журналирования, меток времени.

5. *Отсутствие учёта приоритетов и критичности.* Метрики конфиденциальности, целостности и доступности имеют разный вес для разных систем, однако в SQuaRE они равнозначны. Как показано в [15, 16], для АСУ ТП приоритетом является доступность и целостность, а для государственных информационных систем – конфиденциальность.

6. *Отсутствие уровней безопасности и чётких порогов.* Стандарты SQuaRE не устанавливают градацию безопасности ПО по уровням (низкий/средний/высокий), что затрудняет принятие решений о допуске ПО к эксплуатации на объектах КИИ.

Таким образом, из проведённого обзора однозначно следует, что существующая система метрик безопасности в SQuaRE требует существенной модернизации с учётом российских требований для КИИ, введения новых подхарактеристик, изменения метода агрегирования и установления уровней безопасности. Рассмотрим характеристику «безопасность» в данной системе стандартов подробнее.

Характеристика «безопасность»

в стандартах по качеству ПО

Первый международный стандарт ISO/IEC 9126, содержащий модель качества ПО, появился в системе национальных стандартов России в 1993 г. [6]. Модель качества, представленная в первой части этого стандарта, классифицирует качество ПО в шести основных характеристиках (функциональность, надёжность, практичность, эффективность, сопровождаемость, мобильность), которые, в свою очередь, детализированы подхарактеристиками. Вопрос безопасности (security) в данном стандарте входит в число подхарактеристик первой характеристики качества – функциональности ПО.

В 2001 г. стандарт [6] был развит в систему из четырёх взаимосвязанных стандартов, включающих модель качества, внешние и внутренние метрики, а также метрики качества при использовании. В 2002 г. опубликован стандарт по классификации программных средств [7], в котором для выполнения требований безопасности ПО должно классифицироваться по уровню защиты от несанкционированного доступа, контрольному следу (контрольной суммой, позволяющей судить о целостности) и обеспечению живучести (устойчивости к внешним воздействиям). При этом могут быть установлены дополнительные классы требований защиты, в стандарте приведен пример: слабая, средняя, сильная.

Таким образом, в период с 1993 по 2002 г. в международных стандартах была сформирована трехуровневая модель качества ПО, связанная с

безопасностью информации: безопасность являлась подхарактеристикой функциональности и характеризовалась степенью защиты от НСД, контрольным следом и обеспечением живучести.

В 2015 г. стандарт [6] был заменен на стандарт [9]. По сравнению с версией [6] к основным характеристикам качества ПО как характеристика, а не подхарактеристика функциональности, в данном стандарте была добавлена «безопасность» с подхарактеристиками «конфиденциальность», «целостность», «безотказность», «отслеживаемость» и «подлинность» [10, 11].

Приведем показатели характеристики «безопасность» одного из недавних стандартов SQuaRE [12].

Показатели конфиденциальности (доступность только легальным пользователям):

- контроль доступа,
- корректность шифрования данных,
- сложность криптографических алгоритмов.

Показатели целостности:

- целостность данных,
- защита данных от повреждения,
- защита от переполнения буфера.

Показатели неотказуемости от выполненных действий:

- использование цифровой подписи.

Показатели отчетности:

- полнота журнала действий пользователей,
- хранение системного журнала.

Показатели аутентификации:

• корректность механизма аутентификации (число реализуемых правил к определенным к реализации),

• полнота реализации правил аутентификации (число реализованных к необходимому реализовать).

Анализ приведенных показателей показывает, что перечень подхарактеристик в стандарте нуждается в совершенствовании с точки зрения современного состояния нормативно-правовой базы РФ и теории защиты информации. Если в стандарте [6], который является базовым стандартом по качеству ПО, показатель защищенности был указан в конце перечисления атрибутов из состава характеристики «функциональные возможности», то в настоящее время в условиях непрерывного роста кибератак показатели безопасности и надежности функционирования ПО выходят на первый план. В частности, еще одна подхарактеристика должна характеризовать отсутствие в ПО уязвимостей, НДВ, выполнением требований защиты от НСД и других недостатков (в терминах ГОСТ Р 56939).

Эта подхарактеристика должна отвечать на вопрос о собственно безопасности исходного и исполняемого кода программ. Также, на наш взгляд, необходимо внести уточнения в первый показатель конфиденциальности, показатели целостности и неотказуемости.

В следующем разделе в виде концепции предложена обновленная система подхарактеристик.

Концепция уточнения подхарактеристик безопасности ПО для систем КИИ

При наличии утвержденной МУ основной метрикой предложено считать степень парирования угроз безопасности, связанных с ПО. В отсутствие МУ возможны два подхода к оценке безопасности ПО.

Первый подход состоит в оценке безопасности собственно программы на разных этапах жизненного цикла разработки [2]. Второй подход учитывает возможности внешнего и внутреннего нарушителя использовать уязвимости и другие недостатки ПО с целью НСД к данным и информационным ресурсам системы, для которой это ПО разрабатывалось. При данном подходе количество метрик существенно больше, чем в первом. Реализуем второй подход. Уточнение подхарактеристик безопасности ПО проведем с минимальными изменениями применяемых на практике национальных стандартов системы SQuaRE, но с дополнениями метрик на основе существующих требований нормативно-правовой базы и анализа научных работ.

Унифицированная система подхарактеристик и метрик

Введём шесть подхарактеристик с уникальными индексами метрик, устраняющими путаницу в обозначениях, которая имела место в предшествующих работах.

Конфиденциальность (C)

Для этой метрики п. 8.7 стандарта [11] предлагает следующую функцию измерения для ответа на вопрос «какая часть конфиденциальных элементов данных защищена от НСД?»:

$$X = 1 - A / B, \quad (1)$$

где A – число конфиденциальных элементов данных, доступ к которым разрешен без авторизации; B – число конфиденциальных элементов данных, доступ к которым ограничен.

Такая система оценки, на наш взгляд, не совсем отражает задачу контроля доступа в рамках обеспечения конфиденциальности защищаемой информации, сформулированной в ст. 2 № 149-ФЗ «Об информации, информационных технологиях и о защите информации»: «Конфиденциальность информации – обязательное для выполнения лицом, получившим доступ к определенной информации, требование не передавать такую информацию третьим лицам без согласия ее обладателя». В случае использования функций контроля доступа средствами операционной системы (ОС) защищаемой информацией в процессах разработки является исходный код ПО, для которого формула (1) применима.

В случае встроенных в разрабатываемое ПО механизмов управления доступом в функции измерения подхарактеристики качества «контроль доступа» предлагается учитывать следующие метрики:

C_1 – метрика соответствия реализации встроенной в ПО модели управления доступом утвержденной политике безопасности, измеряемая как отношение реализованных функций к заданным политикой безопасности. Важно отметить, что C_1 оценивает

именно модель управления доступом (разграничение прав субъектов доступа к объектам), в то время как K_1 оценивает наличие МУ безопасности информации в целом. Эти метрики имеют разное смысловое назначение: C_1 отвечает на вопрос «как реализовано разграничение доступа?», а K_1 – на вопрос «какие угрозы учтены при проектировании системы защиты?». МУ (K_1) является исходным документом, определяющим перечень актуальных угроз, а модель управления доступом (C_1) – одной из мер парирования этих угроз. Следовательно, K_1 имеет более высокий приоритет и является заградительной метрикой.

- C_2 – метрика журналирования событий доступа;
- C_3 – периодический аудит прав доступа;
- C_4 – анализ аномальной активности пользователей;
- C_5 – защита от НСД и учётных записей, где

$C_i \in [0,1]$ – значения предложенных метрик.

Безопасность кода ПО (K)

Следующей подхарактеристикой безопасности в стандарте [11] является корректность шифрования данных. В случае использования для защиты информации, содержащейся в информационных системах, шифровальных (криптографических) средств защиты информации должны применяться требования, установленные ФСБ России в соответствии с ч. 5 ст. 16 Федерального закона от 27 июля 2006 г. № 149-ФЗ «Об информации, информационных технологиях и о защите информации», и требования приказа [5].

Для систем объектов КИИ, согласно 187-ФЗ, применение шифрования в явном виде не является обязательным, а зависит от категории объекта и обрабатываемой в системе информации. Для значимых объектов (ЗО) КИИ обязательным является применение сертифицированных на соответствие требованиям по безопасности систем защиты информации и шифрование каналов связи при передаче конфиденциальных сведений из защищенного периметра через сети общего пользования (перечисленные требования обычно выполняются подсистемой защиты информации), а также использование в ЗО КИИ безопасного ПО в соответствии с [1].

Если шифрование данных встроено в прикладное ПО, для оценки корректности шифрования можно воспользоваться функцией измерения данной подхарактеристики из стандарта [11], которая построена на формуле (1). В случае, когда функция шифрования не встроена в разрабатываемое ПО, а выполняется подсистемой безопасности, логично вместо подхарактеристики «корректность шифрования данных» для разрабатываемого ПО ввести подхарактеристику «безопасность кода ПО», поскольку требования [1] построены на парадигме устранения (блокирования) угроз безопасности, связанных непосредственно с разработанным для системы объекта КИИ ПО: наличия в программе уязвимостей, НДВ,

несовершенной защитой от НСД и другими недостатками ПО, могущими повлечь в процессах разработки и эксплуатации реализацию угроз безопасности информации.

В функции измерения подхарактеристики качества «безопасность (security) кода ПО» предлагается учитывать следующие метрики:

- K_1 – наличие утвержденной МУ безопасности информации, связанной с использованием ПО; при составлении перечня угроз безопасности и их описания рекомендуется учитывать положения ГОСТ Р 58412, а также угрозы безопасности информации Банка данных угроз безопасности информации ФСТЭК России, других источников (например, методологии STRIDE, Open Web Application Security Project (OWASP), DREAD и пр.). В МУ рекомендуется указывать использованную при моделировании методологию, в том числе в случае ее собственной разработки. При отсутствии утвержденной МУ коэффициент $K_1 = 0$. Нейтрализация всех выявленных угроз из утвержденной МУ приведет к значению $K_1 = 1$;

- K_2 – с применением SAST выявлены синтаксические ошибки, стилевые нарушения, простые логические дефекты, устранены все известные на момент анализа уязвимости, в том числе находящиеся в банке данных БДУ ФСТЭК России, а также в базах данных SVE, OWASP;

- K_3 – с применением анализаторов состава исходного кода (SCA) в ПО выявлен весь заимствованный код, который может содержать уязвимости, НДВ и другие недостатки ПО;

- K_4 – с применением динамических анализаторов скомпилированного кода, в том числе фаззинг-анализа, выявлены проблемы производительности, утечки памяти, проблемы многопоточности, недостатки безопасности при взаимодействии с окружением;

- K_5 – проведение тестов на проникновение (пентестов) среды разработки с целью выявления уязвимостей инфраструктуры, используемой в процессе создания ПО (системы контроля версий, сборочные стенды, репозитории артефактов, среды тестирования). Результаты пентестов должны быть задокументированы, а выявленные уязвимости устранены или приняты с оформлением акцептованных рисков. При проведении пентестов и устранении критических уязвимостей $K_5 = 1$; при частичном проведении или наличии неустранённых уязвимостей среднего уровня – $0 \leq K_5 < 1$; при отсутствии пентестов – $K_5 = 0$. Важное уточнение: K_5 оценивает безопасность среды разработки, за которую ответствен разработчик ПО. Безопасность среды функционирования (эксплуатационной инфраструктуры заказчика) предлагается оценивать отдельно на этапе приёмочных испытаний с использованием метрики E_1 , которая не входит в предложенную систему и должна

определяться эксплуатантом на основе его модели угроз и политик безопасности.

Такое разделение устраняет конфликт ответственности: разработчик не может и не должен отвечать за безопасность инфраструктуры, которая ему не принадлежит и находится вне его контроля. Здесь $K_i \in [0, 1]$ – значения предложенных метрик. Как показано выше, метрики K_1 и K_2 являются заградительными и учитываются отдельно от агрегированной оценки подхарактеристики Y_K , которая вычисляется по «графируемым» метрикам K_3 , K_4 , K_5 .

Целостность (I)

В функции измерения подхарактеристики качества «целостность» предлагается учитывать следующие метрики:

- I_1 – обеспечение целостности эталонных версий ПО, включающее в себя управление и контроль целостности версий методом вычисления контрольных сумм; при регулярном вычислении и последующей проверке контрольных сумм $I_1 = 1$; при нерегулярном $0 \leq I_1 < 1$;

- I_2 – применение электронной подписи (ЭП) эталонных версий ПО для подтверждения целостности артефактов сборки; при применении ЭП к артефактам сборки $I_2 = 1$, когда ЭП не применяется, $I_2 = 0$;

- I_3 – резервирование эталонных версий ПО; при регулярном резервировании $I_3 = 1$;

- I_4 – контроль целостности данных, обрабатываемых ПО; при регулярном контроле целостности данных $I_4 = 1$;

- I_5 – защита данных от повреждения и модификации, в том числе с помощью минимизации доступа; при отсутствии защиты $I_5 = 0$, при наличии $I_5 = 1$.

Аналогично метрикам K_1 , K_2 метрика I_1 выделена в качестве заградительной и не входит в агрегированную оценку Y_I , которая вычисляется по метрикам $I_2 - I_5$.

Неотказуемость (N)

В стандарте [11] установлена только одна метрика – «использование цифровой подписи». Как показано в работе [43], одной указанной метрики для обеспечения неотказуемости от выполненного действия недостаточно. В функции измерения подхарактеристики качества «неотказуемость» введем следующие метрики:

- N_1 – обеспечение надежной идентификации и аутентификации пользователя согласно стандартам [13, 14]: $N_1 = 1$ при полном выполнении рекомендаций перечисленных стандартов; при неполном $0 \leq N_1 < 1$;

- N_2 – использование меток времени; при регулярном обязательном использовании $N_2 = 1$, в случае, когда метки времени не используются, $N_2 = 0$;

- N_3 – журналирование всех действий пользователей $N_3 = 1$; при ненадлежащем или нерегулярном журналировании $0 \leq N_3 < 1$;

- N_4 – применение электронной подписи (ЭП) для фиксации действий пользователей; в отличие от I_2 , которая оценивает применение ЭП к артефактам сборки, N_4 оценивает применение ЭП к журналам действий пользователей или к подтверждению факта выполнения операций. При применении ЭП для подтверждения действий $N_4 = 1$; при неприменении $N_4 = 0$.

Отчётность (R)

Метрика «полнота журнала действий пользователей», согласно [11], оценивается как $X = A/B$, где A – количество получений доступа, зафиксированных во всех используемых журналах, B – общее количество получений доступа, осуществленных во время тестирования. Вместо этой метрики, согласно современному состоянию теории защиты информации, предлагается введение следующих метрик:

- R_1 – метрика статистики попыток аутентификации пользователей (отношение количества успешных попыток к количеству всех попыток);

- R_2 – метрика статистики изменения привилегий пользователей (отношение количества успешных попыток к количеству всех попыток);

- R_3 – метрика попыток доступа к объектам (отношение количества успешных попыток к количеству всех попыток);

- R_4 – метрика, определенная в [11] как «хранение системного журнала», также определена как отношение действительной продолжительности хранения системного журнала в стабильных хранилищах с ограниченным доступом к журналу к требуемому периоду хранения системного журнала в стабильных хранилищах.

Аутентификация (A)

В стандарте [11] установлены две метрики: «корректность механизма аутентификации (число реализованных правил к определенным к реализации)» и «полнота реализации правил аутентификации (число реализованных к необходимым реализовать)». Предлагается расширить количество метрик показателей аутентификации до четырех:

- A_1 – метрика соответствия организации процесса аутентификации политике безопасности по доступу;

- A_2 – метрика соответствия уровня строгости криптографии, применяемой в процессе аутентификации, требованиям нормативных требований и стандартов;

- A_3 – метрика корректности реализации механизма аутентификации (действительно разные одновременно применяемые факторы, ограничение числа неуспешных попыток аутентификации и др.), изме-

ряемая как количество реализованных правил к количеству определенных к реализации;

- A_4 – метрика полноты реализации правил аутентификации (число реализованных правил к количеству необходимых к реализации).

Двухконтурная модель агрегирования: заградительные и градуируемые метрики

Из 27 предложенных метрик три – K_1 , K_2 и I_1 – представляют собой нормативно обоснованные требования первого уровня (наличие МУ, статический анализ кода без критических уязвимостей, контроль целостности эталонных версий). Для объектов КИИ их невыполнение не может компенсироваться никакими другими показателями. Остальные 24 метрики являются градуируемыми: они описывают степень (полноту, зрелость) выполнения требования и содержательно допускают промежуточные значения без фатальных последствий для итоговой оценки.

Чтобы эти два класса метрик действительно играли разную роль в итоговой оценке, а не только на уровне терминологии, ниже предлагается пересмотренная схема агрегирования, состоящая из двух независимых контуров: контура качества (градуируемые метрики) и контура допуска (заградительные метрики).

Контур качества

Заградительные метрики K_1 , K_2 , I_1 исключаются из подхарактеристик K и I и не участвуют в вычислении Y_K , Y_I : подхарактеристика K вычисляется по метрикам $\{K_3, K_4, K_5\}$, подхарактеристика I – по метрикам $\{I_2, I_3, I_4, I_5\}$. Внутри каждой из шести подхарактеристик по-прежнему используется гармоническое среднее (формула (5)) – это сохраняет устойчивость к маскировке провала внутри одного направления безопасности (например, если из трёх метрик безопасности кода одна равна нулю, подхарактеристика Y_K обнуляется, что непосредственно указывает на наличие критического дефекта в данной подхарактеристике).

Однако на верхнем уровне – при объединении шести подхарактеристик $Y_C, Y_K, Y_I, Y_N, Y_R, Y_A$ в базовую оценку S_0 – гармоническое среднее уже неприменимо: система из шести содержательно разнородных направлений безопасности не должна обнуляться целиком из-за провала в одном второстепенном направлении, если этот провал не относится к нормативно закреплённым заградительным требованиям. Поэтому для верхнего уровня используется взвешенное среднее арифметическое

$$S_0 = \sum_{j=1}^6 w_j Y_j, \quad \sum_{j=1}^6 w_j = 1, \quad (2)$$

где при отсутствии дополнительного обоснования весов для конкретного класса систем рекомендуется $w_j = 1/6$ (равнозначность подхарактеристик, см. ниже разд. «Заградительные метрики»).

Контур допуска

Заградительные метрики образуют независимую бинарную (точнее, полубинарную) проверку допуска

$$G = \mathbb{I}(K_1 > 0) \cdot \mathbb{I}(K_2 > 0) \cdot \mathbb{I}(I_1 > 0) \in \{0, 1\}. \quad (3)$$

Итоговая оценка

Контур допуска действует как жёсткий множитель поверх контура качества:

$$S = G \cdot S_0 = \mathbb{I}(K_1 > 0 \wedge K_2 > 0 \wedge I_1 > 0) \cdot \sum_{j=1}^6 w_j Y_j. \quad (4)$$

Рисунок 1 иллюстрирует эту двухконтурную схему.

Почему это устраняет прежнее противоречие

В формуле (4) обнуление любой из 24 градуируемых метрик снижает соответствующую Y_j (вплоть до нуля, если внутри подхарактеристики нет ни одной положительной метрики), что пропорционально своему весу w_j уменьшает S_0 , но не обнуляет её целиком, если остальные пять подхарактеристик положительны. Обнуление же любой из трёх заградительных метрик K_1, K_2, I_1 обнуляет G и, следовательно, S – независимо от значения S_0 . Дан-

ная схема обеспечивает содержательную, а не только формальную асимметрию между двумя классами метрик.

Рисунок 2 показывает эту асимметрию на числовом примере: провал одной градуируемой метрики (например, $C_3 = 0$, отмечено на левой паре столбцов) снижает S примерно на величину её веса, тогда как провал заградительной метрики (например, $K_1 = 0$, правая пара столбцов) обнуляет S полностью.

Заградительные метрики

1. K_1 – наличие утверждённой модели угроз безопасности информации (МУ).

Обоснование. Согласно требованиям стандарта [1] и методических рекомендаций ФСТЭК России, модель угроз является фундаментальным документом, определяющим перечень актуальных угроз безопасности информации, поверхность атаки, а также обоснование выбора и достаточности мер защиты. Без МУ невозможно корректно определить, какие угрозы должны быть парированы, какие метрики являются приоритетными и какие средства защиты требуются.

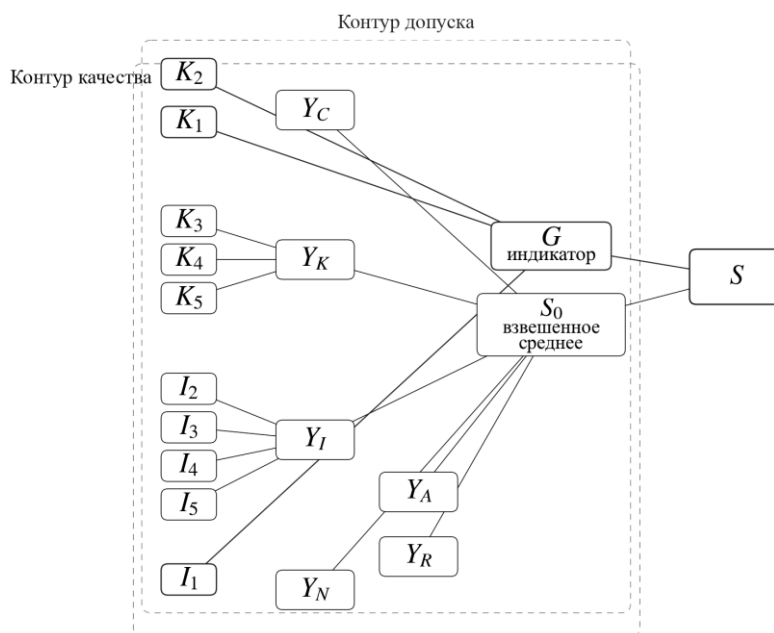


Рис. 1. Двухконтурная схема агрегирования: градуируемые метрики формируют базовую оценку S_0 через гармоническое среднее внутри подхарактеристик и взвешенное среднее между ними; заградительные метрики образуют независимый контур допуска G , действующий как жёсткий множитель

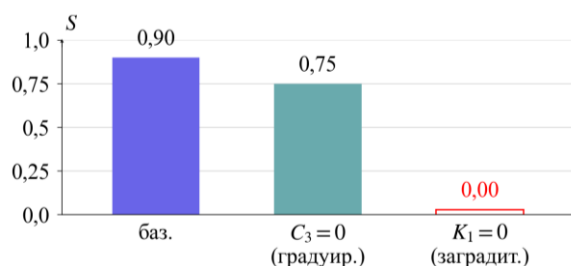


Рис. 2. Реакция итоговой оценки S на провал разных классов метрик при прочих равных: обнуление обычной (градуируемой) метрики снижает S пропорционально её весу, а обнуление любой заградительной метрики обнуляет S полностью

Как показано в работах [15, 16], для критических систем проектирование защиты должно начинаться именно с анализа угроз. Отсутствие МУ означает, что система защиты проектируется «вслепую», что недопустимо для объектов КИИ согласно ст. 11 Федерального закона 187-ФЗ и приказу [3].

2. K_2 – проведение статического анализа исходного кода (SAST) с устранением всех выявленных уязвимостей уровней *Critical* и *High*.

Обоснование. Статический анализ кода является обязательным требованием ГОСТ Р 56939 [1] для всех этапов разработки безопасного ПО. Как пока-

зано в работах [19, 20], именно недостатки кода (уязвимости, НДВ, нежелательные конструкции) являются первичным источником угроз безопасности информации. Отсутствие SAST-анализа или наличие неустранённых критических уязвимостей в коде означает, что ПО содержит заведомо известные недостатки, что делает его неприемлемым для эксплуатации на объектах КИИ. Согласно приказу [3] и п. 5.3 стандарта [1], все выявленные уязвимости должны быть устранены до ввода ПО в эксплуатацию.

3. I_1 – обеспечение целостности эталонных версий ПО (регулярное вычисление и проверка контрольных сумм).

Обоснование. Целостность эталонных версий ПО является критическим требованием для объектов КИИ, поскольку нарушение целостности может быть следствием внедрения вредоносного кода, НДВ или несанкционированных модификаций на этапах разработки, сборки или поставки. Как показано в работе [17], для критических систем целостность ПО является одним из ключевых факторов обеспечения как информационной, так и функциональной безопасности.

Согласно [1], управление конфигурациями и контроль целостности версий являются обязательными требованиями к процессу безопасной разработки. Отсутствие такого контроля делает невозможным гарантировать, что эксплуатируемое ПО соответствует эталонному образцу, прошедшему все проверки.

Правило применения заградительных метрик

В отличие от предыдущей редакции методики, где заградительный статус K_1 , K_2 , I_1 был заявлен, но не выражен в формулах явно (поскольку эти метрики оставались внутри гармонического среднего подхарактеристик Y_K и Y_I наравне с градуируемыми метриками, а это уже само по себе обнуляло всю систему при любом нуле, в формуле (4) заградительные метрики выведены из контура качества и образуют самостоятельный контур допуска G . Это даёт содержательно корректное правило: нулевое значение K_1 , K_2 или I_1 блокирует положительное заключение о безопасности ПО независимо от значений всех остальных 24 метрик, тогда как нулевое значение любой из 24 градуируемых метрик снижает итоговую оценку S пропорционально весу соответствующей подхарактеристики, но не обнуляет её автоматически.

В табл. 1 представлена сводка заградительных метрик.

Унифицированное агрегирование метрик

Для каждой подхарактеристики вычисляется агрегированная оценка $Y_j \in [0,1]$ по формуле гармонического среднего, применяемой к градуируемым метрикам данной подхарактеристики (для K – это $\{K_3, K_4, K_5\}$, для I – это $\{I_2, \dots, I_5\}$, для остальных четырёх подхарактеристик – полный состав их метрик):

$$Y_j = \begin{cases} 0, & \text{если } \exists i: m_i = 0; \\ \frac{n}{\sum_{i=1}^n \frac{1}{m_i}}, & \text{в ином случае,} \end{cases} \quad (5)$$

где n – число градуируемых метрик в подхарактеристике, m_i – значение i -й метрики.

Таблица 1

Заградительные метрики безопасности ПО для объектов КИИ

Метрика	Содержание	Нормативное обоснование	Приоритет
K_1	Наличие утверждённой МУ	ГОСТ Р 56939, Приказ ФСТЭК №239, ст. 11 ФЗ 187-ФЗ	1 (высший)
K_2	Проведение SAST-анализа с устранением уязвимостей Critical и High	ГОСТ Р 56939, п. 5.3	2
I_1	Обеспечение целостности эталонных версий ПО (контрольные суммы)	ГОСТ Р 56939, п.5.4	3

Итоговая оценка безопасности ПО S вычисляется согласно формуле (4) как произведение индикатора допуска G на взвешенное среднее арифметическое шести подхарактеристик S_0 .

Обоснование иерархической свёртки

Применение двухуровневой схемы агрегирования (гармоническое среднее внутри подхарактеристик, взвешенное арифметическое среднее между ними) является сознательным методологическим выбором, а не математической необходимостью. Этот выбор обусловлен тремя соображениями.

1. *Содержательная неоднородность подхарактеристик.* Каждая из шести подхарактеристик (конфиденциальность, безопасность кода, целостность, неотказуемость, отчётность, аутентификация) описывает качественно различный аспект безопасности ПО. Свёртка всех градуируемых метрик в одну оценку без промежуточной группировки математически возможна, но содержательно неоправдана, поскольку стирает границы между различными классами требований.

2. *Равноправность подхарактеристик.* Для объектов КИИ все шесть подхарактеристик являются обязательными в силу требований стандарта [1] и приказа [3]. Ни одна из них не может быть признана второстепенной с точки зрения самого факта необходимости её контроля. Взвешенное среднее с $w_j = 1/6$ обеспечивает равный вклад каждой подхарактеристики в базовую оценку S_0 .

3. *Разный характер требований внутри и между подхарактеристиками.* Метрики внутри одной подхарактеристики (например, K_3, K_4, K_5 – SCA, DAST, пентест среды разработки) являются взаимодополня-

ющими техниками контроля одного и того же аспекта безопасности кода, и провал любой из них действительно не должен маскироваться остальными – отсюда гармоническое среднее. Подхарактеристики же между собой описывают content-независимые направления (конфиденциальность vs. отчётность и т.д.), и полное взаимное сглаживание между ними столь же нежелательно, как и его отсутствие: значимый провал в одном направлении должен ощутимо снижать S_0 (что и обеспечивает вес $w_j = 1/6$ при провале целой подхарактеристики), но не должно полностью обнулять результаты по остальным пяти направлениям, если ни одно из них не относится к нормативно закреплённым заградительным требованиям.

Для количественной оценки поведения новой схемы на данных модельного примера (см. раздел ниже «Модельный пример расчёта») при сценарии после устранения критического дефекта ($K_1 = 1$, $G = 1$) получается $S_0 = 0,861$ (см. табл. 5), что практически совпадает с прежним значением 0,867, полученным по формуле с гармоническим средним верхнего уровня и включением K_1, K_2, I_1 в состав Y_K, Y_I – за счёт исключения K_1, K_2 из Y_K и использования более мягкого агрегирования на верхнем уровне. Такое поведение ожидаемо и корректно: часть чувствительности к провалам теперь сосредоточена именно в контуре допуска G , а не размазана по формуле гармонического среднего верхнего уровня, как это было прежде.

Такой подход обеспечивает:

- высокую чувствительность к низким значениям градуируемых метрик *внутри* каждой подхарактеристики (гармоническое среднее);
- умеренную, пропорциональную весу реакцию S_0 на провал целой подхарактеристики, вызванный градуируемыми метриками;
- полное и безусловное обнуление итоговой оценки S при провале любой из трёх нормативно обоснованных заградительных метрик – независимо от значения S_0 ;
- согласованность методологии на всех уровнях и, что важно, реальное (а не только терминологическое) различие между заградительными и градуируемыми метриками.

Уровни безопасности разрабатываемого программного обеспечения

На основе предложенных метрик безопасности и с учётом требований нормативных документов пред-

лагается введение уровней безопасности разрабатываемого ПО (табл. 2).

Обоснование пороговых значений. Пороговые значения 0,5 и 0,75 выбраны на основе следующего обоснования. В проанализированных нормативных документах [1, 4, 5] отсутствует прямое указание на конкретные числовые пороги 50 и 75% для оценки безопасности ПО. Приведённые значения являются результатом экспертного обобщения практики применения указанных документов, а также сложившейся отраслевой традиции, согласно которой, выполнение менее 50% требований не позволяет сделать вывод о контролируемости процесса разработки, а выполнение более 75% требований соответствует высокому уровню зрелости процессов.

Аналогичные пороговые значения (0,5 и 0,75) используются в оценочных шкалах стандарта [9] для дифференциации уровней качества, а также в методиках оценки зрелости процессов разработки (например, CMMI, SPICE). Таким образом, предлагаемые пороги являются обоснованными по аналогии, а не прямым нормативным предписанием и могут быть скорректированы для конкретных классов систем на основе анализа рисков.

При $S = 0$ оформляется мотивированное заключение о невозможности использования ПО до устранения причин, приведших к обнулению контура допуска G (т.е. до устранения нулевого значения хотя бы одной заградительной метрики).

В зависимости от уровня рисков реализации выявленных угроз безопасности информации количество уровней безопасности и границы их определения могут варьироваться.

Например, для крупных государственных холдингов, делающих первые шаги к обеспечению безопасности разрабатываемого заказного ПО, могут быть введены начальные уровни безопасности. Допускается уточнение пороговых значений интерпретации общей оценки для конкретных классов или категорий значимости объектов КИИ на основе анализа рисков.

Обоснование выбора гармонического среднего

Для обоснования выбора в табл. 3 приведено сравнение гармонического среднего с другими методами на характерных для КИИ наборах метрик (сравнение относится к агрегированию внутри подхарактеристики, где действительно требуется полная чувствительность к нулю).

Таблица 2

Уровни безопасности ПО для объектов КИИ

Значение S	Уровень безопасности	Интерпретация	Нормативное обоснование
$S = 0$	Недопустимо	Использование ПО запрещено до устранения критических дефектов	Заградительные метрики (Приказ № 239, ГОСТ Р 56939)
$0 < S < 0,5$	Низкий	Использование ПО не допускается до устранения критических дефектов	Выполнение < 50% требований (экспертная интерпретация)
$0,5 \leq S < 0,75$	Средний	Допускается с ограничениями, требуется план повышения	Выполнение 50–75% требований, требуется доработка
$0,75 \leq S \leq 1$	Высокий	Допускается для объектов КИИ без ограничений	Выполнение > 75% требований (ISO/IEC 25010, отраслевая практика)

Таблица 3

Сравнение методов агрегирования			
Тип свёртки	Пример: [0; 0,9; 0,9; 0,9]	Пример: [0,5; 0,9; 0,9; 0,9]	Пример: [0,9; 0,9; 0,9; 0,9]
Среднее арифметическое	0,68	0,80	0,90
Медиана	0,90	0,90	0,90
Минимум	0	0,50	0,90
Перемножение	0	0,36	0,66
Гармоническое	0	0,71	0,90

Как видно из табл. 3:

- среднее арифметическое маскирует критический ноль, давая 0,68 при наличии нулевой метрики;
- медиана полностью игнорирует низкие значения;
- минимум избыточно жесткий при отсутствии нуля;
- перемножение чрезмерно наказывает за некритические снижения;
- гармоническое среднее обеспечивает оптимальную чувствительность: обнуляется при наличии нуля, существенно снижается при наличии низкого значения (0,5) и практически не меняется при высоких значениях.

Именно эти свойства делают гармоническое среднее уместным *внутри* подхарактеристики (где метрики – взаимодополняющие проверки одного аспекта), но неуместным на верхнем уровне между подхарактеристиками, что и обосновывает переход к взвешенному арифметическому среднему в формуле (4).

Модельный пример расчёта

Рассмотрим гипотетический проект по разработке прикладного ПО для объекта КИИ. Значения метрик (0–1) получены экспертным путём на основе результатов тестирования и анализа (табл. 4).

Для визуального анализа исходных данных построены типовые карты метрик. Такая форма представления позволяет быстро обнаруживать не только средний уровень выполнения требований, но и локальные провалы внутри отдельных подхарактеристик.

Рисунок 3 показывает пример определения всех 27 метрик для исходного сценария (с $K_1 = 0$): заградительные метрики выделены рамкой, единственная критическая ячейка – K_1 .

Расчёт по формуле гармонического среднего для градуируемых метрик подхарактеристик:

$$Y_C = \frac{5}{1/0,8 + 1/0,9 + 1/0,7 + 1/0,6 + 1/1,0} = 0,774;$$

$$Y_K = \frac{3}{1/0,8 + 1/0,7 + 1/0,9} = 0,792,$$

(по K_3, K_4, K_5 , без учёта заградительных K_1, K_2);

$$Y_I = \frac{4}{1/1,0 + 1/0,8 + 1/0,9 + 1/1,0} = 0,917,$$

(по $I_2, \dots, I_5$, без учёта заградительного I_1);

$$Y_N = \frac{4}{1/1,0 + 1/0,9 + 1/1,0 + 1/0,8} = 0,917;$$

$$Y_R = \frac{4}{1/0,9 + 1/0,9 + 1/0,8 + 1/1,0} = 0,894;$$

$$Y_A = \frac{4}{1/0,9 + 1/0,8 + 1/0,9 + 1/0,9} = 0,873.$$

Контур допуска:

$G = \mathbb{I}(K_1 = 0, 0 > 0) \cdot \mathbb{I}(K_2 = 0, 9 > 0) \cdot \mathbb{I}(I_1 = 0, 9 > 0) = 0$, поскольку $K_1 = 0$.

Таблица 4

Исходные данные для модельного примера					
C	K (в т.ч. K_1, K_2)	I (в т.ч. I_1)	N	R	A
[0,8; 0,9; 0,7; 0,6; 1,0]	[$K_1 = 0, 0$; $K_2 = 0, 9$; 0,8; 0,7; 0,9]	[$I_1 = 0, 9$; 1,0; 0,8; 0,9; 1,0]	[1,0; 0,9; 1,0; 0,8]	[0,9; 0,9; 0,8; 1,0]	[0,9; 0,8; 0,9; 0,9]

C	0,8	0,9	0,7	0,6	1,0
K	0,0	0,9	0,8	0,9	0,9
I	0,9	1,0	0,8	0,9	1,0
N	1,0	0,9	1,0	0,8	
R	0,9	0,9	0,8	1,0	
A	0,9	0,8	0,9	0,9	
	1	2	3	4	5

Рис. 3. Пример определения метрик исходного сценария

Таким образом, отсутствие утверждённой модели угроз ($K_1 = 0$) обнуляет контур допуска G и, следовательно, итоговую оценку $S = G \cdot S_0 = 0$, даже при высоких значениях S_0 по всем шести подхарактеристикам. Это соответствует логике заградительного контроля, когда фундаментальное требование (наличие модели угроз) является обязательным.

Описание результатов расчёта

Расчёт демонстрирует независимую работу двух контуров модели. Контур качества по всем шести подхарактеристикам показывает высокие значения: конфиденциальность – 0,774, безопасность кода (по градуируемым метрикам) – 0,792, целостность (по градуируемым метрикам) – 0,917, неотказуемость – 0,917, отчётность – 0,894, аутентификация – 0,873 (табл. 5).

Таблица 5

Агрегированные оценки подхарактеристик безопасности ПО (контур качества)

Подхарактеристика	Оценка Y_j	Вклад в S_0 ($w_j Y_j$, $w_j = 1/6$)
Конфиденциальность	0,774	0,129
Безопасность кода (градуир.)	0,792	0,132
Целостность (градуир.)	0,917	0,153
Неотказуемость	0,917	0,153
Отчётность	0,894	0,149
Аутентификация	0,873	0,146
S_0 (сумма)		0,861

Несмотря на это, контур допуска $G = 0$ из-за $K_1 = 0$, поэтому итоговая оценка $S = 0$: ПО не может быть допущено к применению на объекте КИИ, неза-

висимо от того, насколько хорошо выполнены остальные 24 метрики.

Дополнительно рассмотрим чувствительность итоговой оценки к устранению критического дефекта.

Если после разработки и утверждения МУ значение метрики K_1 будет повышено до 1,0 (индикатор $\mathbb{I}(K_1 > 0)$ становится равным 1, $G=1$) при сохранении остальных метрик безопасности кода на исходном уровне, оценка подхарактеристики безопасности кода Y_K не меняется (поскольку K_1 более не входит в её расчёт), а итоговая оценка становится равной $S = S_0 = 0,861$. Такой результат переводит проект из состояния недопустимости в высокий уровень безопасности.

Для сравнения приведём также реакцию на провал градуируемой (не заградительной) метрики: если вместо K_1 обнулить, например, $C_3 = 0$ при $K_1 = 1$ (МУ утверждена), то $Y_C = 0$ (гармоническое среднее внутри подхарактеристики C обнуляется), контур допуска остаётся $G = 1$ (поскольку K_1, K_2, I_1 по-прежнему положительны), и итоговая оценка составит

$$S = 1 \cdot \left(0 \cdot \frac{1}{6} + 0,792 \cdot \frac{1}{6} + 0,917 \cdot \frac{1}{6} + 0,917 \cdot \frac{1}{6} + 0,894 \cdot \frac{1}{6} + 0,873 \cdot \frac{1}{6} \right) = 0,732.$$

В табл. 6 сведены все три сценария и наглядно показана искомая асимметрия: провал заградительной метрики блокирует оценку полностью, провал градуируемой метрики – существенно, но не полностью.

Содержательно полученные результаты позволяют выделить три практических вывода.

Во-первых, наиболее сильными сторонами рассматриваемого проекта являются целостность, неотказуемость и отчётность: все три оценки находятся выше 0,89.

Во-вторых, конфиденциальность имеет наименьшее ненулевое значение среди градуируемых подхарактеристик (0,774), поэтому после устранения критического дефекта именно это направление целесообразно рассматривать как ближайший резерв повышения итогового уровня.

Таблица 6

**Сравнение сценариев:
реакция S на разные классы дефектов**

Сценарий оценки	G	S
Исходное состояние: заградительная метрика $K_1 = 0$	0	0,000
После утверждения модели угроз: все заградительные метрики > 0	1	0,861
Провал одной градуируемой метрики ($C_3 = 0$) при исправной модели угроз	1	0,732

В-третьих, двухконтурная модель впервые в данной методике действительно различает последствия провала заградительной и градуируемой метрики:

первое полностью блокирует допуск, второе понижает уровень (в примере – со «высокого» до «среднего»), но не блокирует его автоматически.

С учётом приведённых иллюстраций итоговая интерпретация результата следующая: ПО не может быть допущено к применению на объекте КИИ до устранения критического дефекта, связанного с отсутствием утверждённой МУ (табл. 7). После разработки и утверждения МУ требуется повторная оценка метрики K_1 , пересчёт контура допуска G и повторное вычисление итоговой оценки безопасности S .

Таблица 7

Интерпретация общей оценки безопасности ПО

Диапазон	Уровень	Пример сценария
$S = 0$	Недопустимо	Исходное состояние ($K_1 = 0, G = 0$)
$0 < S < 0,5$	Низкий уровень	Требуется доработка
$0,5 \leq S < 0,75$	Средний уровень	Провал градуируемой метрики C_3 ($S = 0,732$)
$0,75 \leq S \leq 1,0$	Высокий уровень	После устранения K_1 все метрики в норме ($S = 0,861$)

Ограничения метода

Предложенный подход имеет следующие ограничения, которые необходимо учитывать при практическом применении:

1. *Субъективность значений метрик.* Значения метрик $C_i, K_i, I_i, N_i, R_i, A_i$ выставляются экспертами на основе анализа документации, результатов тестирования и аудита кода. Это вносит элемент субъективизма, особенно для метрик, не имеющих чётких количественных критериев (например, C_4 – анализ аномальной активности). Рекомендуются формализация критериев для каждой метрики с указанием пороговых значений (например, $K_2 = 1$ только при покрытии тестами $SAST \geq 80\%$ кода и устранении всех уязвимостей уровнем Critical и High).

2. *Статичность оценки.* Методика даёт точечную оценку на момент проверки, но безопасность ПО меняется в процессе эксплуатации: появляются новые уязвимости, изменяется среда функционирования, обновляются версии компонентов. Требуется периодический пересчёт метрик (например, ежеквартально или после каждого обновления ПО).

3. *Неполнота покрытия для специфических классов ПО.* Набор из 27 метрик не является исчерпывающим для всех классов ПО. Например, для веб-приложений могут потребоваться дополнительные метрики по защите от OWASP Top 10 (инъекции, нарушение аутентификации, чувствительные данные и т.д.), для встраиваемых систем – метрики по защите от аппаратных атак. Предложенная система является базовой и может быть расширена под конкретный класс систем.

4. *Отсутствие весовых коэффициентов приоритетности между подхарактеристиками.* В базовой версии методики все шесть подхарактеристик

считаются равнозначными ($w_j = 1/6$), однако для разных классов систем приоритеты могут различаться. Например, для АСУ ТП приоритетом являются доступность и целостность, а для государственных информационных систем – конфиденциальность. Формула (4) допускает неравномерные веса w_j , но их обоснование требует отдельного анализа рисков для конкретной системы.

5. *Трудозатратность сбора данных.* Для вычисления всех 27 метрик требуется проведение значительного объема работ: статический и динамический анализ кода, анализ состава, тесты на проникновение, аудит документации, проверка журналов и т.д. Для небольших организаций это может быть экономически нецелесообразно.

6. *Зависимость от актуальности баз уязвимостей.* Метрики K_2 и K_3 зависят от актуальности используемых баз данных уязвимостей (БДУ ФСТЭК, SVE, OWASP, CVE). Если базы не обновляются, оценка может не отражать реального уровня безопасности.

7. *Остаточный риск veto-логики внутри подхарактеристик.* Хотя переход к двухконтурной схеме устранил основной практический риск – полное обнуление S из-за одной случайно не измеренной градуируемой метрики, гармоническое среднее внутри каждой подхарактеристики по-прежнему обнуляет Y_j при любом нулевом значении входящей в неё метрики. Если, например, метрика C_4 (анализ аномальной активности) в организации попросту не отслеживается, это законно должно снижать Y_C , но объективно не всегда отличимо от ситуации «метрика не измерялась» от ситуации «требование провалено». Рекомендуется, как и прежде, предусмотреть отдельное состояние «не измерено» (с последующим требованием обязательного измерения до итоговой аттестации), а не автоматически трактовать отсутствие данных как $m_i = 0$.

8. *Разграничение зон ответственности.* Предложенная система метрик оценивает безопасность ПО с точки зрения разработчика. Безопасность инфраструктуры эксплуатации (серверы, сеть, СЗИ, ОС) требует отдельной оценки и не входит в предметную область данной работы.

Рекомендуется при практическом применении методики чётко фиксировать границы ответственности: разработчик предоставляет оценку по предлагаемым метрикам (включая K_5 для среды разработки), а заказчик на основе этих данных и собственного анализа эксплуатационной среды принимает решение о допуске ПО.

9. *Для АСУ ТП и других технологических систем* рекомендуется пересмотреть весовые коэффициенты w_i в формуле (4) с учётом приоритета доступности и целостности над конфиденциальностью, а также предусмотреть дополнительные метрики, оценивающие время восстановления после сбоев и устойчивость к инцидентам.

Заключение

Проведен краткий анализ научных работ и стандартов, определяющих качество ПО применительно к характеристике «безопасность». Показано, что на текущий момент подхарактеристики показателя безопасности ПО как одной из основных характеристик качества ПО, установленные в системе стандартов SQuaRE, нуждаются в совершенствовании с привлечением специалистов по информационной безопасности. Предложена концепция введения системы количественных оценок безопасности ПО и новой подхарактеристики безопасности ПО «уровни безопасности ПО» как показателя качества разрабатываемого ПО.

Предложена простая трехуровневая система формирования уровней безопасности ПО, опирающаяся на двухконтурную модель агрегирования, в которой заградительные и градуируемые метрики играют математически различную функциональную роль.

Предложенная концепция уровней безопасности нуждается в дальнейшей проработке с точки зрения развития теории защиты информации, в частности, уточнения требований и характеристик уровней безопасности [33]. В частности, установлены, но подлежат дальнейшему дополнению и уточнению следующие показатели безопасности ПО:

- обязательность проверки на наличие известных уязвимостей и их устранение;
- отсутствие в ПО НДВ и нежелательных конструкций;
- обеспечение целостности ПО;
- необходимость наличия в ПО функций разграничения доступа;
- наличие в ПО встроенных функций идентификации и аутентификации, различающихся по уровням доверия, установленным в национальных стандартах [7, 9];
- применение в программном обеспечении криптографических функций согласно требованиям при необходимости использования сервисов безопасности на базе инфраструктуры открытых ключей;
- обеспечение функциональной надежности ПО;
- наличие в ПО функций резервирования (архивирования) и восстановления информации.

Предложенная в данной работе система количественных оценок безопасности разрабатываемого ПО для ИС объектов КИИ может быть расширена на другие классы систем в зависимости от решения владельца (оператора). Для каких-то систем обеспечение конфиденциальности, доступности и целостности может иметь разные приоритеты. Например, для некоторых государственных систем приоритеты могут быть выстроены в порядке: доступность, целостность, конфиденциальность. Заданные уровни безопасности для таких систем могут корректироваться весовыми коэффициентами w_j при вычислении S_0 .

В продолжение работы планируется исследовать вопросы оценки остаточных рисков ПО, достигнутого уровня безопасности ПО [34, 35], соотношения требований функциональной безопасности и без-

опасности информации при использовании ПО, а также провести сравнение требований безопасности и надежности ПО.

Литература

1. ГОСТ Р 56939–2024. Защита информации. Разработка безопасного программного обеспечения. Общие требования. – М.: Стандартинформ, 2024.
2. ГОСТ Р 12207–2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. – М.: Стандартинформ, 2010.
3. Об утверждении требований по обеспечению безопасности значимых объектов КИИ Российской Федерации. Утв. приказом ФСТЭК России от 25 декабря 2017 г. № 239 [Электронный ресурс]. – URL: <https://fstec.ru/dokumenty/vse-dokumenty/prikazy/prikaz-fstek-rossii-ot-25-dekabrya-2017-g-n-239> (дата обращения: 09.10.2026).
4. Методический документ ФСТЭК России. Состав и содержание мероприятий и мер по защите информации, содержащейся в информационных системах. Утв. ФСТЭК России 12 апреля 2026 г. [Электронный ресурс]. – URL: <https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-12-aprelya-2026-g> (дата обращения: 09.10.2026).
5. Приказ ФСТЭК России от 11.04.2025 № 117 «Об утверждении Требований о защите информации, содержащейся в государственных информационных системах, иных информационных системах государственных органов, государственных унитарных предприятий, государственных учреждений» (Зарегистрирован в Минюсте России 16.06.2025 № 82619) // Официальный интернет-портал правовой информации. – URL: <http://publication.pravo.gov.ru/document/0001202506170011> (дата обращения: 09.10.2026).
6. ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристика качества и руководства по их применению. – Введ. 1993-12-28. – М.: ИПК «Изд-во стандартов», 2003.
7. ГОСТ Р ИСО/МЭК ТО 12182–2002. Информационная технология. Классификация программных средств. Введ. 2002-06-11. – М.: ИПК «Изд-во стандартов», 2002.
8. ГОСТ Р ИСО/МЭК 25000–2021. Системная и программная инженерия. Требования и оценка качества систем и программных средств (SQuaRE). Руководство. Введ. 2021-10-25. – М.: ФГБУ «РСТ», 2021.
9. ГОСТ Р ИСО/МЭК 25010–2015. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов. Введ. 2015-05-29. – М.: Стандартинформ, 2015.
10. ГОСТ Р ИСО/МЭК 25022–2021. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Измерение качества при использовании. Введ. 2021-10-25. – М.: ФГБУ «РСТ», 2021.
11. ГОСТ Р ИСО/МЭК 25023–2021. Системная и программная инженерия. Требования и оценка качества систем и программной продукции (SQuaRE). Измерения качества системы и программной продукции. Введ. 2021-11-19. – М.: ФГБУ «РСТ», 2021.
12. ГОСТ Р 70921–2023. Системная и программная инженерия. Требования и оценка качества систем и программной продукции (SQuaRE). Концепция требований к качеству. Введ. 2023-09-14. – М.: Российский институт стандартизации, 2023.
13. ГОСТ Р 70262.1–2022. Защита информации. Идентификация и аутентификация. Уровни доверия к идентификации. – М.: Стандартинформ, 2022.
14. ГОСТ Р 70262.2–2025. Защита информации. Идентификация и аутентификация. Уровни доверия к аутентификации. – М.: Стандартинформ, 2025.
15. Шубинский И.Б. Функциональная надежность информационных систем. Методы анализа. – Ульяновск: Печатный двор, 2012. – 296 с.
16. Шубинский И.Б. Надежные отказоустойчивые информационные системы. Методы синтеза. – Ульяновск: Печатный двор, 2016. – 544 с.
17. Шубинский И.Б. Функциональная безопасность систем управления на железнодорожном транспорте / И.Б. Шубинский, Е.Н. Розенберг. – Москва; Вологда: Инфра-Инженерия, 2023. – 360 с.
18. Марков А.С. Модели оценки и планирования испытаний программных средств по требованиям безопасности информации // Вестник МГТУ им. Н.Э. Баумана. – Сер.: Приборостроение. – 2011. – С.1. – С. 90–103.
19. Марков А.С. Систематика уязвимостей и дефектов безопасности программных ресурсов / А.С. Марков, А.А. Фадин // Защита информации. Инсайд. – 2013. – № 3 (51). – С. 56–61.
20. Марков А.С. Эвристический анализ безопасности программного кода / А.С. Марков, В.А. Матвеев, А.А. Фадин, В.Л. Цирлов // Вестник МГТУ им. Н.Э. Баумана. – Сер.: Приборостроение. – 2016. – № 1 (106). – С. 98–111. DOI: 10.18698/0236-3933-2016-1-98-111.
21. Мерзлякова Е.Ю. Обзор методов верификации и оценки качества программного обеспечения / Е.Ю. Мерзлякова, Е.В. Янченко // Вестник СибГУТИ. – 2023. – Т. 17, № 1. – С. 92–106. DOI: 10.55648/1998-6920-2023-17-1-92-106.
22. Кулямин В.В. Технологии программирования. Компонентный подход. – Лекция 5: Качество программного обеспечения и методы его контроля [Электронный ресурс]. – URL: <https://studfile.net/preview/993777/page:14/> (дата обращения: 22.07.2026).
23. Goertzel K.M. Software Security Assurance: A State-of-Art Report (SAR) / K.M. Goertzel, Th. Winograd, H.L. McKinley et al. – Fort Belvoir, VA: Information Assurance Technology Analysis Center, Data and Analysis Center for Software. – 2007.
24. Woody C. Exploring the Use of Metrics for Software Assurance (SEI Report CMU/SEI-2018-TN-004) / C. Woody, R.J. Ellison, Ch. Ryan. – Carnegie Mellon University, USA: Software Engineering Institute, 2020. <https://doi.org/10.1184/R1/12366842.v1>.
25. Платформа Рефлектиз [Электронный ресурс]. – URL: <https://www.reflectiz.com/product/platform-overview> (дата обращения: 21.07.2026).
26. State of Web Exposure 2026 [Электронный ресурс]. – URL: <https://www.reflectiz.com/learning-hub/web-exposure-2026-research>. (дата обращения: 22.07.2026).
27. Systematic literature review of machine learning based software development effort estimation models / S. Wen, Z. Li, Y. Lin, C. Hu, C. Huang // Information and Software Technology. – 2012. – Vol. 54, No. 1. – P. 41–59. DOI: 10.1016/j.infsof.2011.09.002.
28. Naeem S. Predicting Faults in High Assurance Software / S. Naeem, J. Khoshgoftaar, J. Van Hulse // 12th IEEE High Assurance Systems Engineering Symposium, 3–4 November 2010. – San Jose, CA, USA: IEEE, pp. 26–34. DOI: 10.1109/HASE.2010.29.
29. Hassan A.E. The top ten List: Dynamic Fault Prediction / A.E. Hassan, R.C. Holt // Proceedings of the 21st IEEE International Conference on Software Maintenance, 26–29 September 2005. – Budapest, Hungary: IEEE, 2005. – P. 263–272. DOI: 10.1109/ICSM.2005.91.

30. Kindervag J. (2010-11-05). Build Security Into Your Network's DNA: The Zero Trust Network Architecture. Forrester Research [Электронный ресурс]. – URL: https://www.virtualstarmedia.com/downloads/Forrester_zero_trust_DNA.pdf (дата обращения: 22.07.2026).

31. Témolé F. An Integrated Approach to Managing Software Quality in Complex Systems / F. Témolé, D. Ata-nasova // American Journal of Software Engineering and Applications. – 2025. – Vol. 13, No. 1. – P. 1–17. DOI: 10.11648/j.ajsea.20251301.11.

32. Champion K. Qualities of Quality: A Tertiary Review of Software Quality Measurement Research / K. Champion, S. Khatri and B. M. Hill // arXiv preprint. – 2021. – arXiv: 2107.13687. DOI: 10.48550/arXiv.2107.13687.

33. Деркач Е.А. Оценка нулевого доверия на этапах жизненного цикла создания программных продуктов / Е.А. Деркач, А.А. Шелупанов, С.Н. Новиков, Д.С. Брагин // Доклады ТУСУР. – 2025. – Т. 28, № 4. – С. 144–151. DOI: 10.21293/1818-0442-2025-28-4-144-151.

34. Баранов В.В. Методика и алгоритмы расчета защищенности элементов распределенных информационных систем в условиях деструктивного воздействия / В.В. Баранов, А.А. Шелупанов // Доклады ТУСУР. – 2022. – Т. 25, № 4. – С. 88–100. DOI: 10.21293/1818-0442-2022-25-4-88-100.

35. Деркач Е.А. Методологические подходы к обеспечению безопасности программных продуктов / Е.А. Деркач, А.А. Шелупанов // Доклады ТУСУР. – 2024. – Т. 27, № 1. – С. 37–43. DOI: 10.21293/1818-0442-2024-27-1-37-43.

Бочков Александр Владимирович

Д-р техн. наук, учёный секретарь,
АО «Научно-исследовательский и проектно-конструкторский институт информатизации, автоматизации и связи на железнодорожном транспорте» (АО «НИИАС»)
Нижегородская ул., 27, стр. 1, г. Москва, Россия, 109029
ORCID: 0000-0003-0211-8320
Тел.: +7-916-234-40-32
Эл. почта: a.bochkov@vniias.ru

Сабанов Алексей Геннадьевич

Д-р техн. наук, гл. эксперт АО «НИИАС»
Нижегородская ул., 27, стр. 1, г. Москва, Россия, 109029
ORCID: 0000-0002-2231-0240
Эл. почта: a.sabanov@vniias.ru
Тел.: +7-985-924-52-09

Шелупанов Александр Александрович

Д-р техн. наук, член-корр. РАН, проф., президент
Томского государственного университета систем управления и радиоэлектроники (ТУСУР)
Ленина пр-т, 40, г. Томск, Россия, 634050
ORCID: 0000-0003-2393-6701
Тел.: +7 (382-2) 55-55-55
Эл. почта: saa@fb.tusur.ru

Поступила в редакцию: 30.07.2026.

Принята к публикации: 31.08.2026.

Bochkov A.V., Sabanov A.G., Shelupanov A.A.

A Concept for Developing Security Metrics for Application Software in Critical Information Infrastructure Facilities

Relevance. The increasing regulatory and legal requirements for information protection in critical information infrastructure

(CII) facilities necessitate the development of a system of quantitative measures for assessing the security of application software being developed and used. **Purpose of the study.** Development of an advanced method for quantitative security assessment of application software based on the methodology of the international standards for software quality. **Methods.** The study comprehensively employs methods of system analysis, formalization, synthesis, comparative analysis, computational experiment, and regulatory alignment. **Scientific novelty.** A concept has been developed for introducing a system of quantitative security assessments of application software and a new software security sub-characteristic, «software security levels», as a quality indicator for software under development. A three-level framework for establishing software security levels is proposed, based on a two-contour aggregation model in which blocking and gradable metrics play mathematically distinct functional roles. **Results.** The applicability of the proposed concept is demonstrated through a case study of calculating security metrics. **Practical significance.** The proposed system of quantitative assessments for refined security metrics can be applied in projects involving the development and modernization of application software for critical information infrastructure facilities.

Keywords: software security, critical information infrastructure, SQuARE, security metrics, harmonic mean, blocking metrics, threat model, software security levels.

DOI: 10.21293/1818-0442-2026-29-2-160-176

References

1. GOST R 56939–2024. *Zashchita informatsii. Razrabotka bezopasnogo programmogo obespecheniya. Obshchie trebovaniya* [Information protection. Development of secure software. General requirements]. Moscow, Standartinform, 2024 (in Russ.).
2. GOST R 12207–2010. *Informatsionnaya tekhnologiya. Sistemnaya i programmaya inzheneriya. Protessy zhiznennogo tsikla programmnykh sredstv* [Information technology. System and software engineering. Software life cycle processes]. Moscow, Standartinform, 2010 (in Russ.).
3. *Ob utverzhdenii Trebovaniy po obespecheniyu bezopasnosti znachimyykh ob'ektov KII Rossiiskoi Federatsii* [On the approval of Requirements for ensuring the security of significant objects of critical information infrastructure of the Russian Federation]. Order No. 239 of the FSTEC of Russia dated December 25, 2017 (in Russ.). Available at: URL: <https://fstec.ru/dokumenty/vse-dokumenty/prikazy/prikaz-fstek-rossii-ot-25-dekabrya-2017-g-n-239> (accessed: 09 October 2026).
4. *Metodicheskii dokument FSTEC Rossii. Sostav i sodержание meropriyatii i mer po zashite informatsii, soderzhashikhsya v informatsionnykh sistemah* [Methodological document of the FSTEC of Russia. Composition and content of measures and activities to protect information contained in information systems]. Approved April 12, 2026 (in Russ.). Available at: <https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-12-aprelya-2026-g> (accessed: 09 October 2026).
5. Order of the FSTEC of Russia No. 117 dated 11.04.2025 "On Approval of the Requirements for the Protection of Information Contained in State Information Systems and Other Information Systems of State Bodies, State Unitary Enterprises, and State Institutions" (Registered with the Ministry of Justice of Russia on 16.06.2025 No. 82619) // Official Internet Portal of Legal Information. URL: <http://publication.pravo.gov.ru/document/0001202506170011> (accessed: 09.10.2026).
6. GOST R ISO/IEC 9126–93. *Informatsionnaya tekhnologiya. Otsenka programnoi produktsii. Kharakteris-*

tika kachestva i rukovodstva po ikh primeneniyu [Information technology. Evaluation of software products. Quality characteristics and guidelines for their use]. Introduced 1993-12-28. Moscow, «Izdatel'stvo standartov» Publ. house, 2003 (in Russ.).

7. GOST R ISO/IEC TO 12182–2002. *Informatsionnaya tekhnologiya. Klassifikatsiya programmykh sredstv* [Information technology. Classification of software]. Introduced 2002-06-11. Moscow, «Izdatel'stvo standartov» Publ. house, 2002 (in Russ.).

8. GOST R ISO/IEC 25000–2021. *Sistemnaya i programmnaya inzheneriya. Trebovaniya i otsenka kachestva sistem i programmykh sredstv (SQuaRE)*. Rukovodstvo [System and software engineering. Systems and software quality requirements and evaluation (SQuaRE). Guide]. Introduced 2021-10-25. Moscow, Russian Institute of Standardization, 2021 (in Russ.).

9. GOST R ISO/IEC 25010-2015. *Sistemnaya i programmnaya inzheneriya. Trebovaniya i otsenka kachestva sistem i programmnogo obespecheniya (SQuaRE)*. *Modeli kachestva sistem i programmykh produktov* [System and software engineering. Systems and software quality requirements and evaluation (SQuaRE). System and software product quality models]. Introduced 2015-05-29. Moscow, Standartinform, 2015 (in Russ.).

10. GOST R ISO/IEC 25022-2021. *Sistemnaya i programmnaya inzheneriya. Trebovaniya i otsenka kachestva sistem i programmnogo obespecheniya (SQuaRE)*. *Izmerenie kachestva pri ispol'zovanii* [System and software engineering. Systems and software quality requirements and evaluation (SQuaRE). Measurement of quality in use]. Introduced 2021-10-25. Moscow, Russian Institute of Standardization, 2021 (in Russ.).

11. GOST R ISO/IEC 25023–2021. *Sistemnaya i programmnaya inzheneriya. Trebovaniya i otsenka kachestva sistem i programmoi produktsii (SQuaRE)*. *Izmereniya kachestva sistemy i programmoi produktsii* [System and software engineering. Systems and software quality requirements and evaluation (SQuaRE). Measurement of system and software product quality]. Introduced 2021-11-19. Moscow, Russian Institute of Standardization, 2021 (in Russ.).

12. GOST R 70921–2023. *Sistemnaya i programmnaya inzheneriya. Trebovaniya i otsenka kachestva sistem i programmoi produktsii (SQuaRE)*. *Kontseptsiya trebovaniy k kachestvu* [System and software engineering. Systems and software quality requirements and evaluation (SQuaRE). Concept of quality requirements]. Introduced 2023-09-14. Moscow, Russian Institute of Standardization, 2023 (in Russ.).

13. GOST R 70262.1-2022. *Zashchita informatsii. Identifikatsiya i autentifikatsiya. Urovni doveriya k identifikatsii* [Information protection. Identification and authentication. Levels of trust in identification] Moscow, Russian Institute of Standardization, 2022 (in Russ.).

14. GOST R 70262.2-2025. *Zashchita informatsii. Identifikatsiya i autentifikatsiya. Urovni doveriya k autentifikatsii* [Information protection. Identification and authentication. Levels of trust in authentication] Moscow, Russian Institute of Standardization, 2025 (in Russ.).

15. Shubinskii I.B. *Funktsional'naya nadezhnost' informatsionnykh sistem. Metody analiza* [Functional reliability of information systems. Methods of analysis]. Ulyanovsk, Regional Printing House «Pechatnyi dvor», 2012. 295 p. (in Russ.).

16. Shubinskii I.B. *Nadezhnye otkazoustoichivye informatsionnye sistemy. Metody sinteza* [Reliable fault-tolerant information systems. Methods of synthesis]. Ulyanovsk, Regional Printing House «Pechatnyi dvor», 2016, 544 p. (in Russ.).

17. Shubinskii I.B., Rozenberg E.N. *Funktsional'naya bezopasnost' sistem upravleniya na zheleznodorozhnom transporte* [Functional safety of control systems in railway transport]. Moscow, Vologda, Infra-Inzheneriya, 2023, 360 p. (in Russ.).

18. Markov A.S. *Modeli otsenki i planirovaniya ispytaniy programmykh sredstv po trebovaniyam bezopasnosti informatsii* [Models for evaluation and planning of software testing for information security requirements]. *Herald of the Bauman Moscow State Technical University. Series Instrument Engineering*, 2011, Special issue, pp. 90–103 (in Russ.).

19. Markov A.S., Fadin A.A. *Sistematika uyazvimostei i defektov bezopasnosti programmykh resursov* [Taxonomy of vulnerabilities and security defects of software resources]. *Zashita informatsii*. Inside, 2013, No. 3, pp. 56–61 (in Russ.).

20. Markov A.S., Matveev V.A., Fadin A.A., Tsirlov V.L. [Heuristic analysis of source code security]. *Herald of the Bauman Moscow State Technical University. Series Instrument Engineering*, 2016, no. 1 (106), pp. 98–111. DOI: 10.18698/0236-3933-2016-1-98-111 (in Russ.).

21. Merzlyakova E.Yu., Yanchenko E.V. [Review of Software Quality Verification and Evaluation Methods]. *Vestnik SibGUTI*, vol. 17, 2023, no. 1, pp. 92–106. DOI: 10.55648/1998-6920-2023-17-1-92-106 (in Russ.).

22. Kulyamin V.V. *Tekhnologii programmirovaniya. Komponentnyy podkhod. Lektsiya 5. Kachestvo programmnogo obespecheniya i metody ego kontrolya* [Programming Technologies. Component-Based Approach. Lecture 5. Software quality and methods of its control] (in Russ.). Available: <https://studfile.net/preview/993777/page:14/> (accessed: 22 July 2026).

23. Goertzel K.M., Winograd Th., McKinley H.L. et al. *Software Security Assurance: A State-of-Art Report (SAR)*. Fort Belvoir, VA: Information Assurance Technology Analysis Center, Data and Analysis Center for Software, 2007.

24. Woody C., Ellison R.J., Ryan Ch. *Exploring the Use of Metrics for Software Assurance (SEI Report CMU/SEI-2018-TN-004)*. Carnegie Mellon University, USA, Software Engineering Institute, 2020. <https://doi.org/10.1184/R1/12366842.v1>.

25. *Platforma Reflektiz* [Reflectiz Platform] (in Russ.). Available at: <https://www.reflectiz.com/product/platform-overview>, free (accessed: 21 July 2026).

26. *State of Web Exposure 2026*. URL: <https://www.reflectiz.com/learning-hub/web-exposure-2026-research> (accessed: 22 July 2026).

27. Wen S., Li Z., Lin Y., Hu C., Huang C. Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, 2012, 54, no. 1, pp. 41–59. DOI: 10.1016/j.infsof.2011.09.002.

28. Naeem S., Khoshgoftaar J., Van Hulse J. Predicting Faults in High Assurance Software. *Proceedings of the 12th IEEE International Symposium on High Assurance Systems Engineering*, November 3-4, 2010, San Jose, CA, USA, IEEE, pp. 26–34. DOI: 10.1109/HASE.2010.29.

29. Hassan A.E., Holt R.C. The top ten List: Dynamic Fault Prediction. *Proceedings of the 21st IEEE International Conference on Software Maintenance*, 26-29 September 2005, Budapest, Hungary, IEEE, 2005, pp. 263–272. DOI: 10.1109/ICSM.2005.91.

30. Kindervag J. (2010-11-05). *Build Security Into Your Network's DNA: The Zero Trust Network Architecture*. Forrester Research. Available at: https://www.virtualstarmedia.com/downloads/Forrester_zero_trust_DNA.pdf (accessed: 22 July 2026).

31. Témolé F., Atanasova D. An Integrated Approach to Managing Software Quality in Complex Systems. *American Journal of Software Engineering and Applications*, 2025, vol. 13, no. 1, pp. 1–17. DOI: 10.11648/j.ajsea.20251301.11.

32. Champion K., Khatri S., Hill B.M. Qualities of Quality: A Tertiary Review of Software Quality Measurement Research. arXiv preprint, 2021, arXiv: 2107.13687. DOI: 10.48550/arXiv.2107.13687.

33. Derkach E.A., Shelupanov A.A., Novikov S.N., Bragin D.S. [Zero Trust assessment across the software product development lifecycle]. *Doklady Tomskogo gosudarstvennogo universiteta sistem upravleniya i radioelektroniki*, 2025, vol. 28, no. 4, pp. 144–151. DOI: 10.21293/1818-0442-2025-28-4-144-151 (in Russ.).

34. Baranov V.V., Shelupanov A.A. [Cognitive model for assessing the security of information systems for various purposes]. *Doklady Tomskogo gosudarstvennogo universiteta sistem upravleniya i radioelektroniki*, 2022, vol. 25, no. 4, pp. 88–100. DOI: 10.21293/1818-0442-2022-25-4-88-100 (in Russ.).

35. Derkach E.A., Shelupanov A.A. [Domestic and foreign experience in security testing]. *Doklady Tomskogo gosudarstvennogo universiteta sistem upravleniya i radio-elektroniki*, 2024, vol. 27, no. 1, pp. 37–43. DOI: 10.21293/1818-0442-2024-27-1-37-43 (in Russ.).

Alexander V. Bochkov

Doctor of Technical Sciences, Scientific Secretary,
Joint-Stock Company «Research and Design Institute
for Informatization, Automation and Communication
in Railway Transport» (JSC «NIAS»)
27/1, Nizhegorodskaya St., Moscow, Russia, 109029
ORCID: 0000-0003-0211-8320
Phone: +7-916-234-40-32
Email: a.bochkov@vnias.ru

Alexey G. Sabanov

Doctor of Technical Sciences, Chief Expert, JSC «NIAS»
27/1, Nizhegorodskaya St., Moscow, Russia, 109029
ORCID: 0000-0002-2231-0240
Phone: +7-985-924-52-09
Email: a.sabanov@vnias.ru

Alexander A. Shelupanov

Doctor of Technical Sciences,
Corresponding Member of the Russian Academy of Sciences
President, Tomsk State University
of Control Systems and Radioelectronics (TUSUR)
40, Lenin Ave., Tomsk, Russia, 634050
ORCID: 0000-0003-2393-6701
Phone: +7 (382-2) 55-55-55
Email: saa@tusur.ru

Received: 30.07.2026.

Accepted: 31.08.2026.