# BIS JOURNAL

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ БИЗНЕСА

№4 (63) 2026



**Арутюн АВЕТИСЯН**
ИСП РАН
**«ИИ — это инструмент в руках человека»** → 13



**Сергей КИРЮШКИН**
Газинформсервис
**На «кухне» PKI-Форума**
→ 62

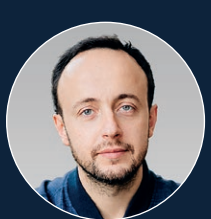
# ИИ: ЧТО ЗА ПОРОГОМ?



**Сетевая безопасность** → 5

**Искусственный интеллект** → 18

**РБПО** → 69



**Рустам ГУСЕЙНОВ**
ПК РАД КОП (radcop.online)
**Как проверить ИИ-агента** → 36



**Карина НАПАДОВСКАЯ**
Лаборатория Касперского
**Вендор в эпоху ИИ** → 76



**Дмитрий КЛЮЧНИКОВ**
CodeScoring.Save
**Современное хранилище артефактов** → 86

# ГОД 2026. ЭПОХА РБПО



**Дмитрий ПОНОМАРЕВ**
советник директора ИСП РАН, преподаватель МГТУ им. Н. Э. Баумана ИУ10

**В завершающем 2026 год номере BIS Journal темы РБПО и ИИ — в том числе их пересечения — вновь находятся в фокусе спецпроектов. Это неудивительно, поскольку обе темы в настоящий момент являются ключевыми для ФСТЭК России на всех уровнях — от разработки национальных стандартов усилиями ТК362 и до принятия конкретных приказов, требований и методик, регулирующих деятельность и развитие отраслей ИБ (и ИТ в целом). Новичкам данный материал может быть полезен в качестве основы для систематизации, опытным специалистам и участникам РБПО-сообщества ФСТЭК России и ИСП РАН — в качестве своеобразного «дайджеста» событий и подборки интересных фактов.**

## 5 ВОПРОСОВ ЭКСПЕРТУ

За время работы в испытательной лаборатории, особенно в период 2024–2026 гг., я вёл список типовых вопросов от организаций, впервые сталкивающихся с сертификацией (продуктов или процессов). На очереди свежая пятёрка:

***— Я могу получить сертификат РБПО и 5 лет обновлять СЗИ самостоятельно хотя бы и каждый день по два раза?***

— Полный «agile» с выкаткой в прод обновлённых версий СЗИ с такой частотой на текущем этапе практически невозможен, даже при наличии сертификата на процессы РБПО [1]. Уточнения приказа 55 «Положение о сертификации» (п. 71.1 [2]) позволяют проводить испытания, связанные с изменением функций безопасности СЗИ, самостоятельно (без привлечения лаборатории), однако не отменяют все остальные положения приказа, в т.ч. п. 74 о необходимости предоставления материалов испытаний во ФСТЭК России.

***— Подтверждение требования реализации процессов РБПО оператором информационной системы (приказ 117 п. 50 [3]) возможно только посредством предъявления сертификата на процессы РБПО?***

— Строго говоря — да. Но за ~2.5 года действия сертификации процессов получено всего 9 сертификатов РБПО, что очевидно на порядки меньше, чем число подлежащих аттестации ГИС. Поскольку данный вопрос является сейчас одним из самых горячих, ответ на него представителей ФСТЭК России звучит примерно так: вводить промежуточных степеней сертификации РБПО не планируется. Необходимо отстраивать процессы РБПО не только качественно, но и «для себя, а не для регулятора». Отстроенные процессы подразумевают и регламенты, и результирующие артефакты, предъявление комплекта которых при прохождении аттестационной процедуры будет являться свидетельством реализации РБПО. Для получения независимой оценки состояния процессов, помощи в их доводке до требуемого уровня и в уточнении регламентов можно воспользоваться услугами экспертов в области РБПО.

***— Требования к сертификации процессов «вообще» и к процессам РБПО при сертификации СЗИ никак не связаны?***

— После ввода в действие 12 мая 2026 г. актуальной версии Методики ВУ и НДВ [4] ситуация изменилась. Пункт 1.3 Методики явно рекомендует применять положения Методики при реализации процессов по ГОСТ Р 56939–2024. Методика — в отличие от ГОСТ (и семейства ГОСТ РБПО в целом) — раскрывает конкретные алгоритмы и вводит численные константы для различных видов исследований. Сопоставление положений Методики с требованиями ГОСТ можно увидеть, например, здесь [5]. «Рекомендации» Методики — на мой взгляд — являются тем типом рекомендаций, которые по своей сути ближе к полноценному требованию, и должны трактоваться примерно так: если вы реализовали процесс «по сути» качественно, с очевидной пользой для себя, то положения Методики являются рекомендациями. Если же процесс реализован «для вида», с целью поскорее получить заветный сертификат, тогда нереализация очевидно полезных, лучших практик, сконцентрированных в Методике, весьма вероятно вызовет много вопросов.

***— Мне сказали, что я могу быстро и легко пройти сертификацию РБПО — главное подготовить 25 регламентов (а их можно и LLM-кой сгенерировать)?***

— В корне неверное утверждение, см. как минимум объём сертификационных проверок, раскрытый в п. 19 [1], а также любое из выступлений руководителя Органа по сертификации процессов РБПО ИСП РАН Вартана Падаряна (например, [6][7] или найти в поисковике самое све-

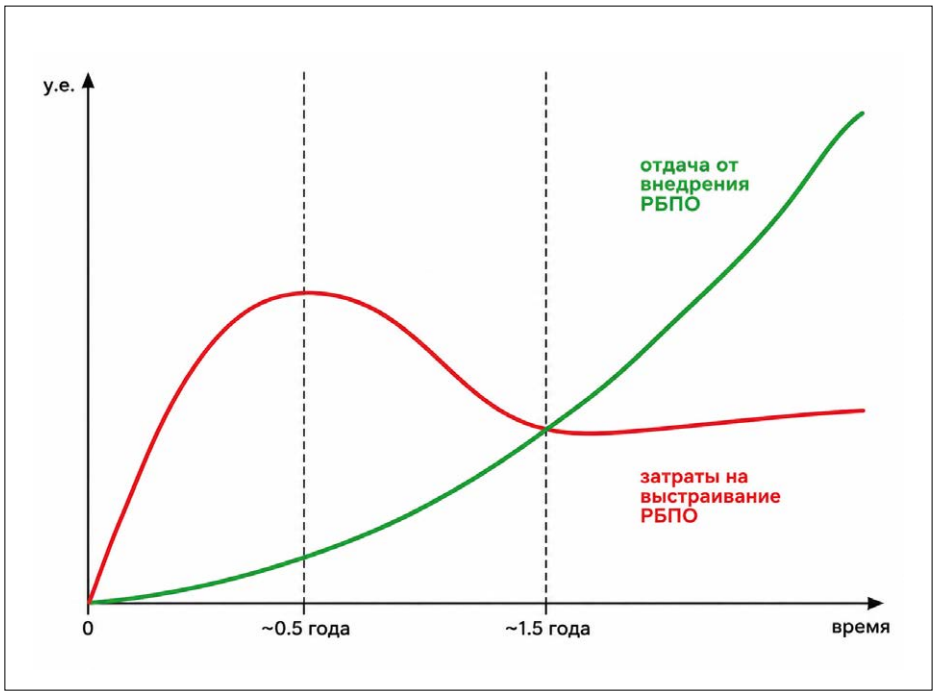



**Рисунок 1.** В средне-долгосрочной перспективе внедрение РБПО всегда приводит к значимому положительному экономическому эффекту

жее). По личному опыту общения с более чем половиной прошедших (или — готовящихся пройти) сертификацию компаний, можно однозначно утверждать: сертификация РБПО — это инженерно-насыщенный процесс, фокус которого на практическом ознакомлении с реально работающими процессами и конкретными людьми, процесс реализующими. Отделаться бумагами не получится ни при каких обстоятельствах, а фейковые (в т.ч. «неактуализированные») регламенты только поспособствуют досрочному прекращению процедуры.

***— Когда, по вашему опыту, стоит задуматься о выходе на процедуру сертификации РБПО?***

— Приведу любимый график, сформированный на основе собственного опыта. Чем раньше вы начнёте внедрять (развивать) РБПО, тем лучше для качества и безопасности продукта, эффективности бизнеса в целом, и тем ближе момент готовности к сертификации. В то же время внедрение редко занимает менее 1.5 года (от околонулевого состояния), а начальный этап требует существенных ресурсовложений — не только на закупку инструментов и выделение рабочего времени на «наведение порядка», но и на смену «культуры» компании, что порой достаточно болезненно. Однако в средне-долгосрочной перспективе внедрение РБПО всегда приводит к значимому положительному экономическому эффекту (рис. 1).

## РБПО.РФ

Портал РБПО.рф создан как часть проекта «Унифицированная среда разработки безопасного отечественного ПО ФСТЭК России» и развивается ИСП РАН с целью объединения сообщества инженеров и руководителей, заинтересованных во внедрении процессов качественной и безопасной разработки. Лучший совет, если вы ещё не посещали портал, — зайти на него и ознакомиться лично. Особого внимания заслуживают следующие разделы и материалы портала:

- Главная страница с лентой новостей и событий.
- Раздел «УНС», освещающий наработки проекта по созданию Унифицированной среды РБПО. Проект должен способствовать популяризации и доступности РБПО-подхода для некрупных компаний, а также может быть основой для создания отечественных коммерческих конвейеров РБПО (один из примеров реализации рассмотрен в следующем материале номера [8]).
- Раздел «РБПО», в настоящий момент включающий два подраздела:
  - примеры шаблонов регламентов процессов РБПО (создаются ИСП РАН по опыту работы Органа по сертификации в сотрудничестве с представителями отрасли);
  - карта отечественных или открытых инструментов РБПО, сопоставленная с классификатором процессов по ГОСТ Р 56939–2024.
- Раздел «Блоги», в котором публикуются оригинальные материалы, подготовленные компаниями — участниками РБПО-сообщества ФСТЭК России и ИСП РАН. Особая ценность раздела в том, что все материалы проходят рецензирование силами коллектива ИСП РАН и привлекаемых экспертов. Среди множества замечательных материалов не могу не выделить «Шпаргалку для методолога», подготовленную коллегами из PGPro [9].
- Раздел «Экспертам», наиболее ценный для тех, кто выстраивает у себя процессы РБПО, так как он содержит проекты типовых документов по безопасной разработке, которые могут быть адаптированы и утверждены в организациях в качестве локальных нормативных актов. Материалы основаны на опыте практического внедрения РБПО и учитывают положения отраслевых стандартов. Документы публикуются по мере готовности и согласования с экспертами ФСТЭК России и будут охватывать все ключевые процессы РБПО.

Приглашаем заинтересованных авторов и экспертов к публикации и дискуссии, а также ждём ваших предложений по развитию ресурсов портала.

## СТАНДАРТЫ, НОРМАТИВКА, ПЕРСПЕКТИВЫ

Работа над развитием регуляторики ведётся активно, с привлечением сформировавшегося и пополняемого сообщества специалистов различных профилей, искренне увлечённых своим делом. Далее я подсвечу несколь-

ко наиболее значимых и интересных лично для меня проектов:

- ГОСТ «Композиционный анализ. Общие требования». Разрабатывался с 2022 (ФСТЭК России, КодСкоринг, ИСП РАН, НТЦ Фобос-НТ). Голосование по проекту стандарта было запущено 12 октября и прошло успешно, документ передан в Росстандарт, текст стандарта доступен к ознакомлению [10]
- проект ГОСТ «Технологии динамического анализа. Общие требования». Материалы по проекту стандарта выложены на публичное обсуждение на портале РБПО.рф 12 октября, готовится запуск процедуры обсуждения в подкомитете [10]
- проект ГОСТ «Безопасность сборочной среды». Ведётся обсуждение редакции в рабочей группе, материалы встречи РБПО-сообщества ФСТЭК России и ИСП РАН в Санкт-Петербурге 5 ноября, посвящённой обсуждению вопросов безопасности сборочных инфраструктур, доступны в чатах сообщества [11]
- проект ГОСТ «РБПО реализующего технологии ИИ. Общие требования». Активно дорабатывается в настоящий момент, по итогам первого раунда обсуждения в подкомитете (получено более 900 замечаний и предложений) [12].
- проект Требований ФСТЭК России к СЗИ систем ИИ. В настоящий момент текст требований находится в стадии разработки. Известно, что требования относятся к различным аспектам защиты СЗИ, использующих технологии ИИ, либо защищающих их, и будут включать в себя как достаточно классические РБПО-практики (УПД к API модели, минимизация прав доступа к исполняемым файлам и файлам весов и т.п.), так и различные требования к анализу входов/выходов модели, анализу конфигураций, промптов, правил агентских систем, требования к квотированию запросов, а также к контролю (мониторингу) поведения модели. Краткая информация о содержании требований представлена публично Ириной Гефнер (ФСТЭК России) в рамках сессии на ITSEC2026 [13].

## РБПО ИИ и ИИ в РБПО

Вопросам ИИ в номере посвящён целый спецпроект, ключевой в котором, на мой взгляд, является статья Дениса Турдакова (Руководитель центра доверенного ИИ, ИСП РАН), в которой наш коллектив постарался осветить весь спектр вопросов, связанных с доверием к ИИ: от фундаментальных научных проблем до конкретных технологий и продуктовых решений, создаваемых ИСП РАН и участниками Консорциума исследований безопасности технологий ИИ [14]. В дополнение к данному материалу хочу привести ссылку на небольшое, но очень интересное выступление Андрея Белеванцева, посвящённое в том числе истории и задачам создания ГОСТ РБПО ИИ [15], а также ещё раз обратить внимание аудитории на выступления Ирины Гефнер, Дмитрия Евдокимова, а также два круглых стола в рамках [13].

* * *

**В завершение материала совместно с редакцией BIS Journal и Медиа Группой «Авангард» приглашаем посетить главное ежегодное отечественное научно-практическое событие в области ИТ, РБПО и ИИ — Открытую конференцию ИСП РАН — которая пройдёт 09–11 декабря 2026 на площадке МИСиС практически в центре Москвы, где вы сможете не только посетить выставку технологий, научно-практические секции и круглые столы, но и лично встретиться и обсудить вопросы с авторами материалов номера.**

## ИСТОЧНИКИ


[1] https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/poryadok-provedeniya-sertifikatsii-utverzhden-prikazom-fstek-rossii-ot-1-dekabrya-2023-g-n-240
[2] https://fstec.ru/dokumenty/vse-dokumenty/prikazy/prikaz-fstek-rossii-ot-3-aprelya-2018-g-n-55
[3] https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/trebovaniya-utverzhdeny-prikazom-fstek-rossii-ot-11-aprelya-2025-g-n-117
[4] https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-12-maya-2026-g
[5] https://cs.groteck.com/IB_4_2026/68/
[6] https://www.tbforum.ru/2026/program/information-security
[7] https://www.isprasopen.ru/2025/ (секция «Разработка безопасного ПО: качество, доверие, сообщество»)
[8] Статья «Конвейеризация РБПО — ключевой фактор роста передового ИТ-бизнеса» на с. 72
[9] https://secure-software.ru/blog/pgpro-rukovodstvo-po-bezopasnoy-razrabotke-po-shpargalka-dlya-metodologa
[10] https://t.me/sdl_community/11417
[11] https://t.me/sdl_arch
[12] https://fstec.ru/tk-362/standarty/proekty/proekt-natsionalnogo-standarta-gost-r-18
[13] https://www.itsecexpo.ru/2026/program/local-ai-intelligence-security
[14] Статья «Доверенный ИИ: от научной проблемы к промышленной платформе» на с. 26
[15] https://t.me/ispras/2004

# КОНВЕЙЕРИЗАЦИЯ РБПО — КЛЮЧЕВОЙ ФАКТОР РОСТА ПЕРЕДОВОГО ИТ-БИЗНЕСА



**Дмитрий СОРОКИН**
заместитель генерального директора по производству Базис



**Дмитрий ПОНОМАРЕВ**
соавтор статьи, советник директора ИСП РАН, преподаватель МГТУ им. Н. Э. Баумана ИУ10

**ПАО «Базис» является лидером отечественного рынка в области создания и внедрения инфраструктурного ПО – средств виртуализации, контейнеризации, организации программно-управляемых сетей и предоставления системного ПО «как сервис». Успехи компании всё более заметны и выражены в скорости и качестве создания и развития продуктовой линейки, а выход компании на фондовый рынок – следствие высокого уровня зрелости внутренних процессов. Одним из ключевых аспектов, позволяющих держать темп создания бизнес-ценностей, является приверженность компании парадигме разработки безопасного ПО (РБПО), поддержка и апробация РБПО-подходов, продвигаемых ФСТЭК России, а также автоматизация, конвейеризация практик РБПО. Накопленный компанией опыт было решено обобщить, дополнить инструментами и методологией ИСП РАН и представить результат в виде продукта – конвейера РБПО.**

## ОСНОВА ЛИНЕЙКИ ПРОДУКТОВ «БАЗИС»

В стек инфраструктурных продуктов «Базиса» входят инструменты управления динамической инфраструктурой, платформа создания виртуальных рабочих мест, конвейер для организации полного цикла разработки, решения для организации программно-определяемых сетей и хранилищ, средства защиты виртуальных сред и др. Вместе они образуют единую экосистему проприетарных разработок компании, которые закрывают актуальные потребности заказчиков.

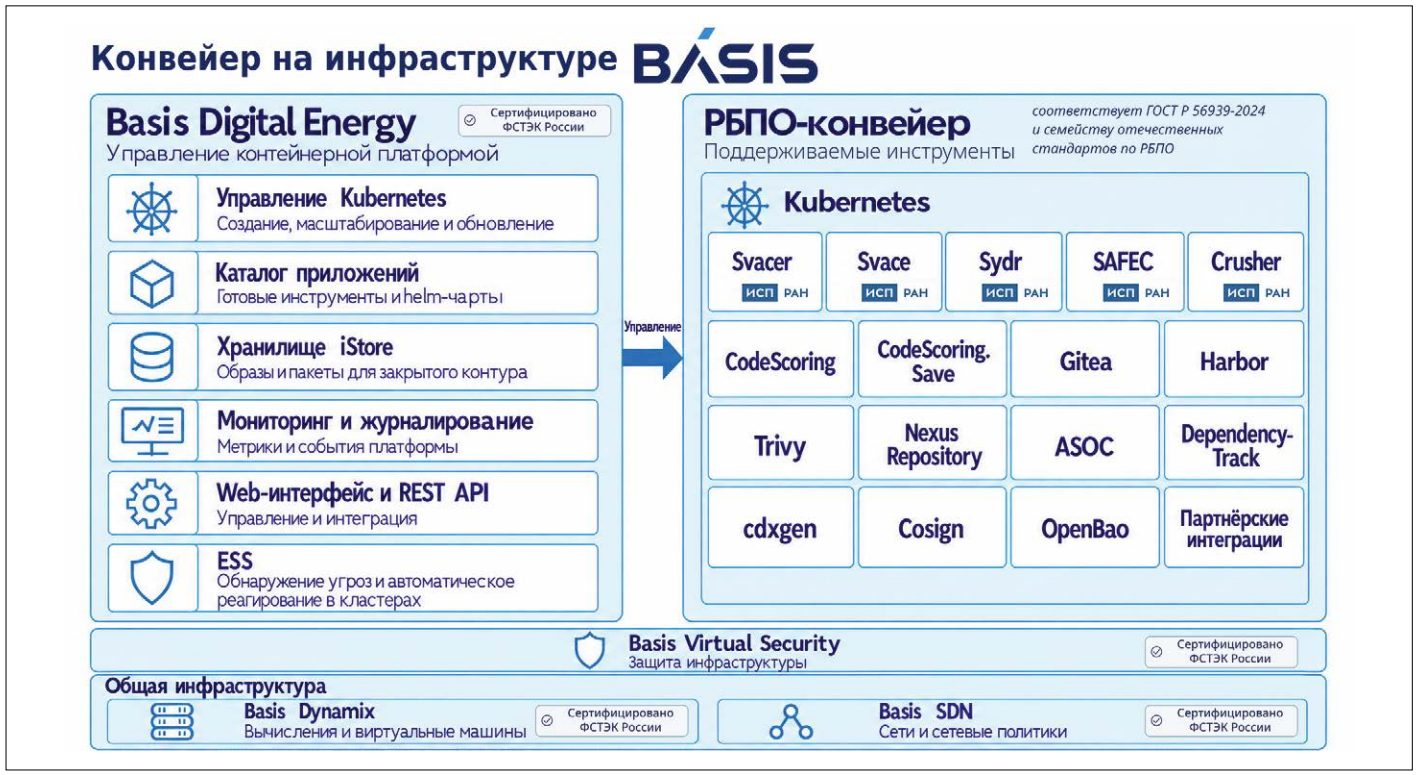
Основу стека составляют высокопроизводительная платформа виртуализации Basis Dynamix Enterprise, решения для защиты сред виртуализации Basis Virtual Security и платформа управления контейнерами Basis Digital Energy. Именно при разработке этих продуктов компания впервые начала применять продвигаемые регулятором подходы РБПО, что в дальнейшем привело к получению ими сертификатов ФСТЭК России и регулярному успешному обновлению этих сертификатов.

## РБПО-КУЛЬТУРА «БАЗИС»

Парадигма РБПО начала активно внедряться в повседневные практики «Базиса» в 2022 году, в период активной трансформации компаний, и к настоящему моменту стала основой, «культурным кодом» разработки. На этом сложном, но критически важном для построения современного, высокоэффективного ИТ-бизнеса пути нам помогают наши друзья и коллеги из ИСП РАН [1] – не только разработчики передовых инструментов анализа, но и ведущие отечественные методологи – а также коллектив испытательной лаборатории НТЦ Фобос-НТ [2]. Мы начинали с апробации инструментов РБПО (используем стек инструментов ИСП РАН, CodeScoring и ряд иных открытых инструментов для решения частных задач анализа), частичного аутсорса анализа кодовой базы и за четыре года дошли до зрелого состояния. Сегодня существенная часть различных видов анализа автоматизирована, основные продукты развиваются в культуре готовности к прохождению сертификационных испытаний, работа в Центре исследования безопасности системного ПО ведётся в отношении нескольких десятков открытых компонент из 4 кластеров [3], а взаимоотношения с ИСП РАН давно вышли за рамки закупок инструментов и услуг и сейчас сосредоточены на исследовательской деятельности в области конвейеризации РБПО и MLSecOps. В качестве давнего участника РБПО-сообщества ФСТЭК России и ИСП РАН мы регулярно выступаем на ведущих отечественных тематических конференциях и с удовольствием делимся опытом [4][5][6]. Конвейеризация процессов композиционного, статического и различных видов динамического анализа кода, в сочетании с их чёткой регламентацией и привлечением технологий ИИ для ускорения рутинных задач и приоритизации анализа выявленных потенциальных дефектов, на наш взгляд, является единственно возможным системным ответом на возрастающие объём и сложность (а также – с развитием вайб-кодинга – определённую «безответственность») подлежащего анализу и использованию кода.

## РБПО-КОНВЕЙЕР «БАЗИС» И ИСП РАН V1.0.BETA

В 2025 году на Kazan Digital Week ПАО Ростелеком и ИСП РАН подписали соглашение о сотрудничестве, направленном на создание ин-




новационных решений в области РБПО [7]. Ключевая задача — создание конвейера РБПО, соответствующего требованиям ГОСТ Р 56939–2024 и семейству расширяющих стандартов в области РБПО, разрабатываемых в ТК 362 [8]. Конвейер разрабатывается как on-premise продукт, и — в перспективе, после окончания внутреннего пилотирования — как облачный сервис. Важное свойство конвейера — изначальная, по умолчанию, синхронизация с проектом Унифицированной среды разработки отечественного безопасного ПО [9].

Ключевые характеристики процесса создания конвейера: изначальное соответствие требованиям стандартов РБПО, приоритет отечественных инструментов исследования и прототипирование API, форматов взаимодействия и контроля инструментов с целью достижения «качественных» эффектов в скорости развёртывания, анализе и управляемости, а также разработка эксплуатационных регламентов в соответствии с эскизами, размещёнными на РБПО.рф [10].

Важный принцип, которым мы руководствуемся при создании конвейера — принцип реализма. Мы считаем, что наиболее верной стратегией является поэтапное создание продукта: движение от базовых элементов инфраструктуры конвейера к базовым инструментам анализа, а затем наращивание инструментального наполнения и кросс-инструментальных возможностей с учётом обратной связи от пользователей и развития регуляторики. При создании такого комплексного продукта необходимо сразу же задумываться о его методическом обеспечении: от регламентов применения до сайзингов, от оценки совместимости с популярными инструментами до создания совместных «коробочных» продуктов. Дорожная карта развития конвейера на среднесрочную перспективу будет представлена вместе с первым публичным релизом. Важную роль в «конвейеризации» инструментов и создании методологии развёртывания и применения конвейера играют коллеги из Фобос-НТ, имеющие обширный опыт выстраивания процессов РБПО, в т.ч. с целью подготовки к сертификации конкретных продуктов — средств защиты информации (СЗИ). Участие опытной испытательной лаборатории позволит уже на этапе первого релиза получить продукт, позволяющий качественно закрывать ряд требований к процессам и артефактам РБПО, обязательных при проведении сертификационных испытаний.

* * *

**Уважаемые коллеги и партнёры, приглашаем вас посетить стенд ПАО «Базис» на главном ежегодном отечественном научно-практическом событии в области ИТ и РБПО — Открытой конференции ИСП РАН 09–11 декабря 2026 [11], где вы сможете лично ознакомиться со стендом и методологией конвейера!**

## ИСТОЧНИКИ


[1] https://www.ispras.ru
[2] https://fobos-nt.ru
[3] https://portal.linuxtesting.ru
[4] https://rbpo2026.avangardpro.ru/materials
[5] https://www.tbforum.ru/2026/program/rbpo
[6] https://habr.com/ru/companies/basis/articles/1002948
[7] https://www.company.rt.ru/press/news/d474813
[8] https://fstec.ru/tk-362/standarty
[9] https://secure-software.ru/uns
[10] https://secure-software.ru/rbpo
[11] https://www.isprasopen.ru



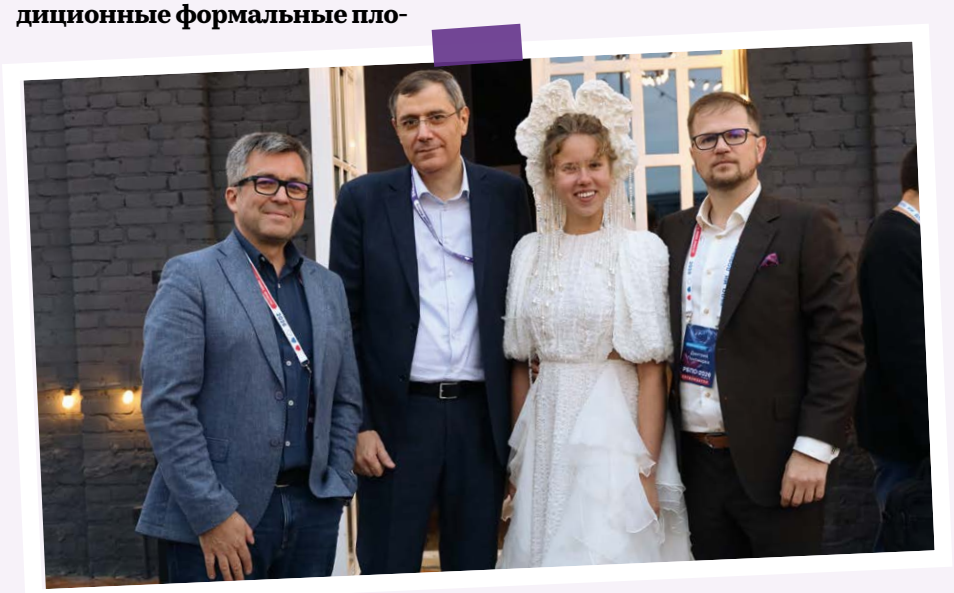
# ПОНЕДЕЛЬНИК НАЧИНАЕТСЯ В СУББОТУ

## 14 АВГУСТА 2026 ГОДА. ГОРЬКО!

***Ещё братья Стругацкие в своей бессмертной книге рассказали о неугомонных людях науки, для которых работа, выходные и праздники сливаются в одно целое. Похоже, эти удивительные люди и организации никуда не делись, и BIS Journal со своей стороны с радостью и удивлением побывал на одном из их замечательных мероприятий.***

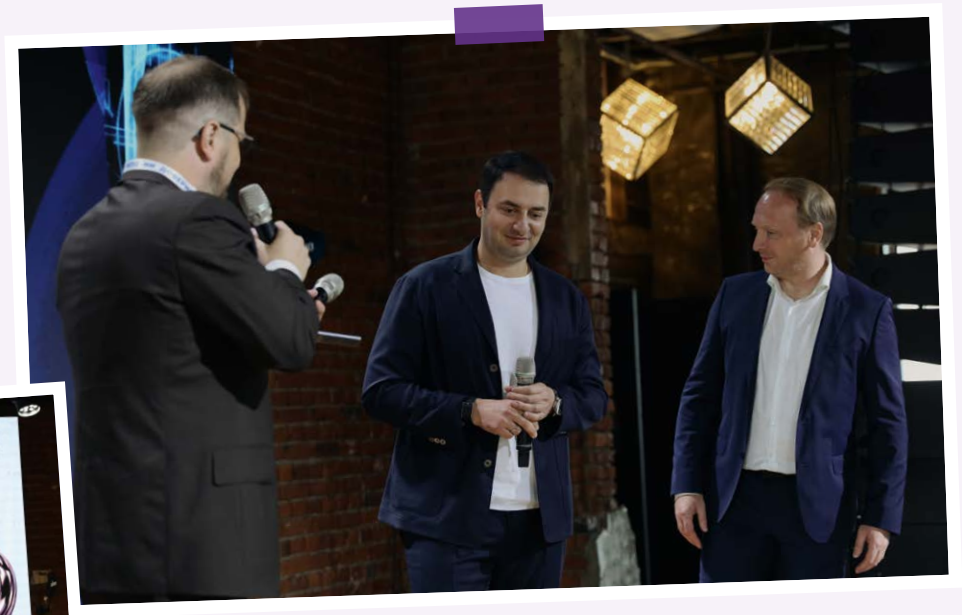
**Александр АБРАМОВ**
BIS Journal

**Участникам Сообщества РБПО ФСТЭК России и ИСП РАН хорошо известны традиционные формальные площадки для обсуждения вопросов на стыке технологий, безопасности и регуляторики, а именно ежегодные отечественные конференции: ТБ Форум, Kaspersky Certification Day, Открытая конференция ИСП РАН и др. В августе этого года появился совершенно нетипичный повод для сбора Сообщества РБПО — свадьба Дмитрия и Елены Пономаревых, которая состоялась 14 августа (пятница). Получившее соответствующее название мероприятие — «Свадьба-конференция РБПО» — представляло собой необычный формат: профессиональная конференция, естественным образом перетекающая в праздник.**



Программа мероприятия была составлена таким образом, чтобы каждый участник конференции смог получить максимум полезного контента, находясь на любом квадратном метре площадки. В одном зале расположилась выставка технологий: Axel, CodeScoring, НПЦ КСБ, BaseALT, Basis, Ideco, Swordfish Security, Deckhouse Флант, СберТех, SALT. BOX — передовые отечественные IT-организации и известные участники Сообщества РБПО. В другом зале об-

суждались вопросы безопасной разработки программного обеспечения и доверенного искусственного интеллекта. За большинством докладов просматривался вопрос: как сделать безопасную разработку естественной частью процесса создания программного обеспечения, а не дополнительной нагрузкой для разработчиков и бизнеса.






Особенно органично звучали темы про коммуникацию участников процесса, обучение специалистов, ответственность руководителей и изменение подходов к организации бизнес- и управленческой модели процесса разработки продуктов.

Арутюн Аветисян, академик РАН, директор ИСП РАН, рассказал о перспективах технологий обеспечения доверия и безопасности ПО, в том числе в эпоху ИИ.

Давид Мартиросов и Дмитрий Сорокин, Ростелеком Basis, рассказали о том, почему РБПО должно перестать восприниматься как дополнительная активность и стать обязательной частью жизненного цикла создания и поставки ПО.

Алексей Смирнов, основатель CodeScoring, рассказал про безопасную разработку и её инфраструктуру в эпоху ИИ.

Давид Мэгтон, сооснователь и технический директор Флант, поделился опытом определения баланса скорости разработки и безопасности, автоматизации и зрелости процессов и рассказал о том, как не превратить РБПО в тормоз для бизнеса.

Дмитрий Евдокимов, основатель Luntry, предметно описал, почему реактивная безопасность уже не успевает за изменениями и как меняется защита в эпоху ИИ.

Александр Гадай и Андрей Иванов, Swordfish Security, разыграли живой диалог на тему поиска компромиссов между разработкой и ИБ.

Всеслав Соленик, директор по кибербезопасности СберТех, и Екатерина Гордеева, психолог и IT-специалист, рассказали о важности умения IT- и ИБ-специалистов разговаривать друг с другом.

Андрей Кузнецов и Елена Пономарева, НТЦ «Фобос-НТ», описали многогранную деятельность современной Испытательной лаборатории.

Юлия Грачёва и Егор Погорелко, МГТУ им. Н. Э. Баумана, рассказали о подготовке специалистов, практическом обучении и вопросе, откуда брать людей, способных заниматься РБПО завтра.

Завершил деловую программу Иван Панченко, астрофизик и сооснователь Postgres Professional, рассказом о самых мощных взрывах во Вселенной и о причастности PostgresPro к развитию астрофизической науки.

Все доклады отлично отразили атмосферу мероприятия: встречи Сообщества РБПО — это не только место получения актуальной информации на профильные темы, но и возможность поговорить с интересными людьми о вещах далеко за пределами привычной повестки.

На «Свадьбе-конференции» в одном пространстве оказались руководители разработки и ИБ, технические директора, генеральные директора, основатели и владельцы компаний, исследователи и практики IT-безопасности. За фуршетными столами, на летней террасе и в зонах отдыха обсуждались вопросы сотрудничества организаций, развития отрасли, вкладов конкретных людей — всё то, на что, как правило, во время обычной конференции остаётся лишь пять минут после доклада. Такова специфика встреч Сообщества РБПО ФСТЭК России и ИСП РАН: здесь можно не только прослушать актуальные доклады на волнующие темы, но и встретиться с людьми, с которыми захочется продолжить разговор.



# МЕСТО ВСТРЕЧИ ИЗМЕНИТЬ НЕЛЬЗЯ

## КАК ВЕНДОРУ ПОДГОТОВИТЬСЯ К ИИ-ОХОТЕ ЗА УЯЗВИМОСТЯМИ С ПОМОЩЬЮ ГЕРМЕТИЧНОЙ СБОРКИ



**Карина НАПАДОВСКАЯ**
руководитель Центра сертификации и соответствия стандартам АО «Лаборатория Касперского»

**Искусственный интеллект (ИИ) превратил поиск уязвимостей из штучной экспертной работы в конвейер. Первыми под удар попадают два аспекта: legacy-код (который десятилетиями считался надёжным лишь потому, что его никто не трогал) и сторонние библиотеки, из которых собраны современные продукты. Разберёмся, как в новых условиях отличить порядочного вендора и почему герметичная сборка становится вопросом не удобства инженеров, а доверия заказчика.**

### СТАРЫЙ КОД НЕ БЫЛ БЕЗОПАСНЫМ — ОН БЫЛ НЕПРОВЕРЕННЫМ

У legacy-кода долгое время была своеобразная презумпция невиновности: если код передаётся от версии к версии продукта, работает пятнадцать лет и в нём ничего не нашли, значит, он надёжен. ИИ эту презумпцию отменил.

Показательный пример — январский выпуск OpenSSL 2026 года, одной из самых изучаемых криптографических библиотек в мире. Все 12 закрытых в нём уязвимостей нашла компания AISLE с помощью собственной ИИ-системы. По оценке исследователей, часть ошибок оставалась незамеченной больше четверти века. Самая серьёзная, CVE-2025-15467, в зависимости от платформы может привести к удалённому выполнению кода.

Принципиально здесь не то, на что способна одна конкретная ИИ-система, а то, как быстро дешевеет сама технология. Стоимость обнаружения уязвимости падает и для защитников, и для атакующих — у старого кода больше нет «срока давности». Модуль, который не трогали, потому что «работает», — не актив, а отложенный долг, и проценты по нему теперь может начислить кто угодно.

### ЧУЖОЙ КОД — НАШ РИСК

Второй фронт — заимствованные компоненты, и здесь картина ещё тревожнее. От года к году число уязвимостей в них заметно растёт. И всё чаще их находит не человек, а ИИ.

В 2024 году команда Google OSS-Fuzz с помощью сгенерированных большими языковыми моделями фазинг-целей нашла в OpenSSL уязвимость CVE-2024-9143. По оценке исследователей, она существовала около двадцати лет и не могла быть обнаружена существовавшими фазинг-целями, написанными людьми. В апреле 2026 года Anthropic представила Project Glasswing: модель Claude Mythos Preview нашла ошибку 27-летней давности в OpenBSD и 16-летний дефект в FFmpeg — в строке, которую автоматические тесты проходили более пяти миллионов раз. К концу мая, по данным компании, 50 организаций — участников программы сообщили о более чем 10 000 найденных уязвимостей высокого и критического уровня.

Для вендора вывод простой: заказчик покупает продукт целиком, а не «наш код плюс чужой код». Фраза «это уязвимость в сторонней библиотеке» уже давно не работает как оправдание и не снимает ответственности за устранение. А поскольку ИИ ищет уязвимости прежде всего в широко используемых библиотеках, это запускает волну обновлений во всех продуктах, которые их используют.

### ПРАВОПОРЯДОК ПО ЖЕГЛОВУ

У Глеба Жеглова есть фраза, которую я считаю лучшей формулой зрелой безопасной разработки: «Правопорядок в стране определяется не наличием воров, а умением властей их обезвреживать». Перенесём её на нашу отрасль: уровень безопасности продукта определяется не наличием в нём уязвимостей, а умением вендора их находить, устранять и доводить обновления безопасности до потребителей. Продукт без уязвимостей — такая же фикция, как город без преступности.

Отсюда важное следствие: количество опубликованных *Common Vulnerabilities and Exposures* (CVE) — плохая мера качества. Ядро Linux закономерно возглавляет рейтинги по числу CVE. Высокая «регистрируемая преступность» нередко говорит о том, что полиция работает. Вендор, у которого уязвимостей «нет», — либо не ищет, либо молчит, и оба варианта плохие.

Правда, в отличие от самого Жеглова, методы здесь имеют значение. Подброшенный кошелёк в нашей от-

расли — это тихий патч без уведомления регулятора или «улучшение стабильности» в описании релиза вместо честного слова «уязвимость». Цель не оправдывает средства: доверие строится на прозрачном процессе, а не на результате любой ценой.

По каким признакам, по моему мнению, можно отличить порядочного вендора:

**Ищет сам и раньше других**. В конвейер встроены инструменты статического, динамического и композиционного анализа, встроены ИИ-технологии, направленные на поиск уязвимостей. Найти уязвимость самим всегда дешевле, чем узнать о ней от конкурентов или исследователей в рамках bug bounty.

**Знает, из чего состоит продукт**. На вопрос «затронуты ли наши продукты?» после публикации новой уязвимости отвечает за часы, а не за месяцы.

**Реагирует по регламенту, а не по настроению**. Сроки по устранению привязаны к критичности, применимости в конкретных условиях и конкретных информационных системах, выстроены процессы по уведомлению заказчиков и регуляторов.

**Управляет мощностью**. Поток находок будет и дальше кратно расти, и соблюдение регламента по работе с уязвимостями должно быть задачей планирования, а не героизма. При планировании ресурсов уже нельзя ориентироваться на поток сообщений об уязвимостях двухлетней давности.

**Уровень безопасности продукта определяется не наличием в нём уязвимостей, а умением вендора их находить, устранять и доводить обновления безопасности до потребителей. Продукт без уязвимостей — такая же фикция, как город без преступности**

## ЧТО ИМЕННО МЫ ОТГРУЗИЛИ ЗАКАЗЧИКУ?

Сертификация — это проверка конкретной версии конкретного продукта. Заказчик покупает не абстрактный «сертифицированный продукт», а уверенность в том, что у него установлено именно то, что проверили испытательная лаборатория, орган по сертификации и регулятор.

Регулятор стремительно движется в эту сторону. Приказ ФСТЭК России № 9 от 20 января 2026 года сделал перечни заимствованных компонентов с открытым кодом и образов контейнеров обязательными артефактами, которые подают и проверяют при сертификации, обязал лаборатории оценивать их полноту и риски, потребовал уведомлять ФСТЭК России об изменениях в перечне открытых компонентов в течение пяти календарных дней. Непредставление таких сведений стало основанием для приостановки сертификата соответствия. В мае 2026 года вышла и новая методика выявления уязвимостей и НДВ, опирающаяся на ГОСТ Р 56939–2024, которая также расширила объём проверок.

Теперь посмотрим на типичную сборку. Один и тот же проект, собранный в понедельник и в пятницу, может дать разные бинарные файлы: не зафиксированы версии зависимостей, подтянулась свежая минорная версия транзитивной зависимости, на сервере сменился кэш. Злого умысла нет, но контрольная сумма уже не совпадает с формуляром, перечень компонентов не соответствует действительности, а «проверенное» и «отгруженное» разошлись. Хуже, если в момент сборки из реестра подтянулся скомпрометированный пакет: тогда атака на цепочку поставок попадает прямо в сертифицированный продукт.

## ГЕРМЕТИЧНАЯ СБОРКА: СОСТАВ ПОД КОНТРОЛЕМ

Принцип герметичной сборки — заранее определить все библиотеки, которые должны войти в продукт, зафиксировать исходный код, конфигурации, компиляторы и необходимые параметры окружения. Сборка проходит в изолированной среде без доступа к интернету и окружению разработчика: всё берётся из внутреннего хранилища проверенных компонентов. Если компонента нет в хранилище, его хэш не совпадает с зафиксированным или сборка обращается к чему-то необъявленному, она останавливается. Что это даёт на практике:

**Правдивый перечень компонентов**. Software Bill of Materials (SBOM) определён разработчиком до сборки, а не угадывается сканером постфактум.

**Воспроизводимость**. Лаборатория может пересобрать продукт и сравнить контрольную сумму с формуляром, заказчик — сверить полученный дистрибутив.

**Контроль цепочки поставок**. Новый пакет не попадёт в продукт в обход процедуры допуска во внутреннее хранилище.

**Точечные обновления**. Когда в библиотеке находят уязвимость, сразу понятно, какие продукты и версии затронуты, изменение в перечне точечное, а пятидневное уведомление регулятора становится рутиной, а не авралом.

## МЕСТО ВСТРЕЧИ ИЗМЕНИТЬ НЕЛЬЗЯ

Любой заказчик вправе задать вендору три вопроса. Из чего состоит продукт? Каким образом и как быстро вы доведёте до меня устранение уязвимости? Можете ли доказать, что у меня установлен именно тот продукт, который сертифицирован? У порядочного вендора на все три вопроса есть доказательная база, а не пустые обещания.

Встреча с уязвимостями, в своём legacy-коде или в сторонних библиотеках, неизбежна, ИИ лишь ускорил её. Место встречи изменить нельзя. Выбрать можно другое: прийти на эту встречу подготовленным и уйти с неё победителем.

# ВКАЛЫВАЮТ РОБОТЫ… СЧАСТЛИВ ЧЕЛОВЕК?

## КАК ВСТРОИТЬ ИИ В БЕЗОПАСНУЮ РАЗРАБОТКУ ПО



**Егор ИЗОТОВ**
старший консультант отдела методологии и консалтинга ГК «Солар»

**Искусственный интеллект (ИИ) уже стал рабочим инструментом разработки программного обеспечения (ПО). По данным исследования ГК «Солар» и УЦСБ, 82% организаций используют ИИ для разработки ПО. Генеративные модели снижают порог входа в разработку: для части задач пользователю достаточно описать требования на естественном языке. Агентные инструменты способны не только генерировать код, но и участвовать в сборке приложения, функциональном тестировании, подготовке документации, развёртывании и т.д.**

Масштаб изменений показывает и исследование Checkmarx. По его данным, около 49% кода в продуктивных системах опрошенных организаций уже генерируется с помощью ИИ.

Однако функциональный код не обязательно безопасен. В обучающих данных моделей могут присутствовать устаревшие, небезопасные или просто неудачные примеры кода. Вторая проблема связана с пользовательским запросом: он может вообще не содержать требований информационной безопасности (ИБ).

Показательный пример — задача загрузки изображений. Пользователь скорее попросит ИИ просто «сделать программу загрузки картинок», чем перечислит в запросе ещё и требования к проверке Multipurpose Internet Mail Extensions (MIME)-типа, ограничению размера файла и генерации случайного имени. Если требования безопасности не заданы, нельзя рассчитывать, что модель обязательно учтёт их сама. Именно поэтому в руководстве «WIZ. Claude Code security best practices» сказано: «Основой безопасного программирования с использованием LLM является использование явных и подробных инструкций по безопасности в пользовательском запросе».

При вайбкодинге — разработке ПО по описанию задачи на естественном языке — эта проблема становится особенно заметной. Пользователь формулирует нужные функции приложения, но без экспертизы в разработке и ИБ ему сложно самостоятельно подготовить полноценное техническое задание, включающее в себя, помимо функциональных требований, ещё и нефункциональные, такие, как, например, требования безопасности, иные условия и ограничения.

Отсюда возникают две задачи. Сначала исходный запрос нужно преобразовать в полноценное техническое задание. Затем на его основе нужно построить управляемый процесс разработки на базе ИИ, который контролирует безопасность приложения от формирования требований до промышленной эксплуатации.

Автоматизировать при этом нужно именно процесс. Если разработка представляет собой набор несвязанных действий, ИИ лишь автоматизирует и усиливает существующий хаос.

### БАЗОВЫЕ ПРАВИЛА ИИ-КОНВЕЙЕРА

Разработка безопасного программного обеспечения (РБПО) — это совокупность процессов, реализуемых на всех этапах жизненного цикла ПО, направленных на предотвращение появления, выявление и устранение недостатков и уязвимостей программы.

При включении ИИ в этот процесс действуют несколько базовых правил.

Первое касается полномочий. ИИ-агент нужно рассматривать как привилегированного участника разработки и часть поверхности атаки. Для него действуют принцип минимальных прав, подтверждение необратимых действий, полный аудит и возможность немедленной остановки.

Второе правило — разделение ролей моделей. Модель, которая генерирует код, не должна быть единственным механизмом проверки его безопасности. Эту задачу выполняет другой ИИ-контур со своей моделью, системными ограничениями и критериями приёмки.

Третье правило сохраняет финальное решение за человеком. ИИ может подготовить требования, архитектуру, код, результаты анализа или исправление. Ответственность и «право подписи» остаются у специалиста.

Отдельно нужно защищать сам ИИ-контур. Его компрометация даёт злоумышленнику возможность вмешаться в цепочку поставки ПО ещё до выпуска приложения.

На этих правилах строится полный конвейер разработки. ИИ применяется не только для генерации кода. Он участвует в обеспечении безопасности на каждом этапе жизненного цикла. Специализированные агенты получают отдельные роли, ограниченные полномочия и доступ только к необходимым инструментам.

### ЭТАП 1

**Формирование требований и разработка технического задания (ТЗ).** Процесс начинается с пользовательского промпта — описания задачи на естественном языке. Пользователь может не обладать экспертизой в информационной безопасности. В таком случае его запрос, скорее всего, не бу-

дет содержать полный набор формализованных требований ИБ.

Специализированный агент выполняет роль Product Manager или Security Requirements Engineer. Он преобразует исходный запрос в структурированный перечень требований, включая требования безопасности.

Для этого агент использует:

- навык (skill) с базой знаний OWASP Application Security Verification Standard (ASVS) и правилами сопоставления бизнес-требований с требованиями безопасности;
- сервер Model Context Protocol (MCP), который предоставляет агенту доступ к стандартам безопасности, например OWASP ASVS.

Подобная автоматизация уже используется на практике. Мультиагентная система MAS-SRE преобразует бизнес-требования в прослеживаемые требования безопасности. В её архитектуре работают десять специализированных агентов на четырёх этапах. Система сочетает моделирование угроз по STRIDE с генерацией, дополненной поиском (Retrieval-Augmented Generation, RAG). В исследовании она показала 98,6% покрытия при сопоставлении угроз и 87,6% — при сопоставлении контролей.

На этом же этапе работает агент Threat Modeler. Он получает от Product Manager архитектурное описание и выполняет моделирование угроз. В качестве методологии может применяться STRIDE. Агент формирует перечень угроз, оценивает риски и добавляет результат в техническое задание как раздел «Модель угроз».

Полученное ТЗ становится основой для проектирования безопасной архитектуры.

## ЭТАП 2

**Проектирование безопасной архитектуры.** Агент в роли Architect или AppSec Engineer проектирует архитектуру приложения на основе ТЗ. Требования безопасности учитываются уже на этом этапе.

В предлагаемом конвейере агент использует:

- Secure-Design-Patterns — базу знаний безопасных архитектурных паттернов для аутентификации, авторизации, управления сессиями, шифрования, разделения привилегий и других механизмов;
- MCP-сервер architecture-linter, который проверяет архитектурные решения на соответствие политикам безопасности организации;
- MCP-сервер threat-model-store — хранилище моделей угроз для проверки полноты их покрытия.

Один из ключевых элементов конвейера — Governance Contracts. Это машиночитаемые YAML-артефакты с организационными правилами, политиками безопасности и архитектурными стандартами. Контракты передаются последующим агентам.

Так ограничения становятся структурными, а не только декларативными. Они кодируются на уровне инструментов и доступных действий, а не только задаются в промпте.

Следующий этап получает два основных входных параметра — техническое задание и архитектурное решение.

## ЭТАП 3

**Генерация кода.** Агент в роли разработчика фактически выполняет функцию генератора кода. Он получает ТЗ и архитектурное решение, после чего создаёт код приложения.

Для этого предусмотрены:

- Secure-Code-Generation — навык с безопасными паттернами для используемого языка разработки;
- MCP-сервер code-generation, который предоставляет доступ к библиотекам, фреймворкам и шаблонам с предварительно проверенными паттернами безопасности;
- MCP-сервер dependency-resolver для анализа и выбора внешних зависимостей с проверкой их безопасности.

Для критических участков кода сохраняется ручная проверка. Это относится, например, к аутентификации, авторизации и обработке платёжных данных. Такой код должен проверить специалист. Pull request требует его одобрения до слияния.

После генерации приложение переходит к многоуровневой проверке. Выбранные зависимости также анализируются отдельным потоком, уже независимо от механизма их первоначального выбора.

## ЭТАП 4

**Поиск, классификация и исправление уязвимостей.** Этот этап включает несколько параллельных потоков анализа. Каждый использует свой метод, специализированного агента и набор инструментов. Результаты разных проверок затем можно сопоставлять и перекрёстно валидировать, также с использованием инструментов ИИ.

### АНАЛИЗ СТОРОННИХ КОМПОНЕНТОВ И СОСТАВА ПО — OSA/SCA

Агент Dependency Auditor анализирует внешние компоненты и зависимости.

Он использует:

- SCA-Scanning для запуска инструментов анализа состава ПО. Приоритизация учитывает оценки CVSS и EPSS, а также наличие уязвимости в каталоге CISA KEV;
- MCP-сервер sca-registry, который предоставляет доступ к NVD, OSV, GitHub Advisory Database и другим базам уязвимостей зависимостей.

### СТАТИЧЕСКИЙ АНАЛИЗ КОДА — SAST

Агент Code Auditor выполняет статический анализ исходного кода (SAST).

В его инструментарий входят:

- SAST-Scanning для запуска SAST-инструментов и сопоставления результатов с категориями CWE и соответствующими материалами OWASP;
- MCP-сервер code-facts для индексации и запроса фактов о коде с помощью CodeQL-совместимых запросов.

Данные из code-facts позволяют отслеживать потоки данных и разбирать ложноположительные срабатывания.

### ДИНАМИЧЕСКИЙ АНАЛИЗ — DAST

Агент Dynamic Tester выполняет динамический анализ (DAST) работающего приложения.

Он использует:

- DAST-Scanning с параметрами, применимыми к конкретному типу приложения;
- MCP-сервер target-environment, который управляет тестовой средой

и инструментами, а также собирает результаты DAST-проверок.

При необходимости к конвейеру подключаются отдельные потоки фаззинг-тестирования, анализа безопасности контейнеров и мобильных приложений. В фаззинг-тестировании ИИ может использоваться для оптимизации мутаций тестовых входных данных.

Результаты независимых проверок затем агрегируются и передаются на классификацию.

## КЛАССИФИКАЦИЯ И ПРИОРИТИЗАЦИЯ

Все выявленные уязвимости агрегирует агент Vulnerability Classifier.

Он использует:

- CWE-Classification для сопоставления находок с категориями CWE;
- MCP-сервер vulnerability-ledger — единый реестр находок с механизмом глубокого слияния записей.

Vulnerability-ledger поддерживает согласованность данных, в том числе после сброса контекста агента. После агрегации находки классифицируются и приоритизируются.

Для этой задачи особенно важна специализация модели. ГК «Солар» проверила шесть LLM на 20 проектах на языках Java и Python. Объём каждого проекта превышал 100 тыс. строк кода. В отдельных исследованных сценариях публичные LLM пропускали 40–50% уязвимостей.

Специализированная модель DerTriage показывает более высокую точность. В ИИ-плагине Solar appScreener точность автоматической верификации достигает 95%.

Классифицированные и приоритизированные находки передаются отдельному агенту для исправления.

## ИСПРАВЛЕНИЕ И ПРОВЕРКА ИСПРАВЛЕНИЙ

Исправления формирует агент в роли Remediation Engineer. Он использует навык Automated-Remediation.

Агент генерирует исправление и сопровождает его объяснением. В комментарии к коду указываются суть уязвимости и предлагаемое изменение.

Подобный сценарий реализуют, например, GitLab Agentic SAST Vulnerability Resolution и Google CodeMender. Такие инструменты автоматически генерируют исправления и могут подготавливать готовые к слиянию merge request.

Аналогичный сценарий реализован в Solar appScreener. DerCodeFix генерирует контекстные патчи для подтверждённых уязвимостей и объясняет внесённые изменения. В исследовании «Солара» точность модели составляла 78,2% для Java и 83,1% для Python. В актуальной версии продукта точность кодфикса достигает 85%.

Генерация патча и его проверка разделены. MCP-сервер patch-validator должен подтвердить, что исправление устраняет исходную уязвимость. Он также проверяет, не появились ли регрессии.

После проверки исправлений приложение переходит к заключительному контролю перед выпуском.

## ФИНАЛЬНОЕ ТЕСТИРОВАНИЕ И КОНТРОЛЬ ВЫПУСКА

На заключительном этапе агент Security Champion проводит комплексную проверку.

Для этого он использует:

- Security-Sign-Off для финальной проверки всех артефактов безопасности;
- MCP-сервер release-gate, который блокирует выпуск при наличии неустранённых уязвимостей критического и высокого уровней.

Порог критичности для блокировки можно настроить.

Отдельный агент GRC Analyst проверяет соответствие приложения установленным требованиям управления, рисков и соответствия (Governance, Risk and Compliance, GRC).

Он использует:

- Compliance-Mapping для сопоставления результатов сканирования с требованиями применимых стандартов и нормативных документов, например приказов ФСТЭК России;
- MCP-сервер audit-evidence для сбора и хранения доказательств соответствия. По сути, это хранилище материалов аудита соответствия.

Таким образом, в разработанном полном цикле разработки под контролем ИИ пользовательский запрос сначала превращается в набор требований и модель угроз. Затем формируются архитектура и код, а после этого приложение проходит независимые виды анализа, классификацию и исправление уязвимостей. Завершает процесс контроль безопасности и соответствия перед выпуском.

## ПОЧЕМУ ОДНОГО ПРОМПТА ДЛЯ ПОИСКА УЯЗВИМОСТЕЙ НЕДОСТАТОЧНО

Эту архитектуру можно сравнить с более простым сценарием. Готовый код передаётся модели с запросом: «Найди в этом коде все уязвимости и устрани их».

У такого подхода есть принципиальное ограничение. Безопасность становится постфактум-проверкой уже созданного приложения. В предлагаемом конвейере контроль начинается раньше и проходит через весь жизненный цикл.

Первое преимущество — сдвиг влево (shift left). Требования ИБ появляются ещё при анализе исходной задачи. Модель угроз и архитектурные ограничения формируются до написания кода. Поэтому часть потенциальных уязвимостей можно предотвратить ещё на этапе проектирования.

Позднее исправление обходится значительно дороже. По данным «Солара», стоимость устранения уязвимости возрастает примерно в десять раз на поздних этапах разработки. На этапе запуска приложения рост может достигать 640 раз. Если уязвимость уже привела к инциденту в массовом сервисе, показатель может достигать 1000 раз.

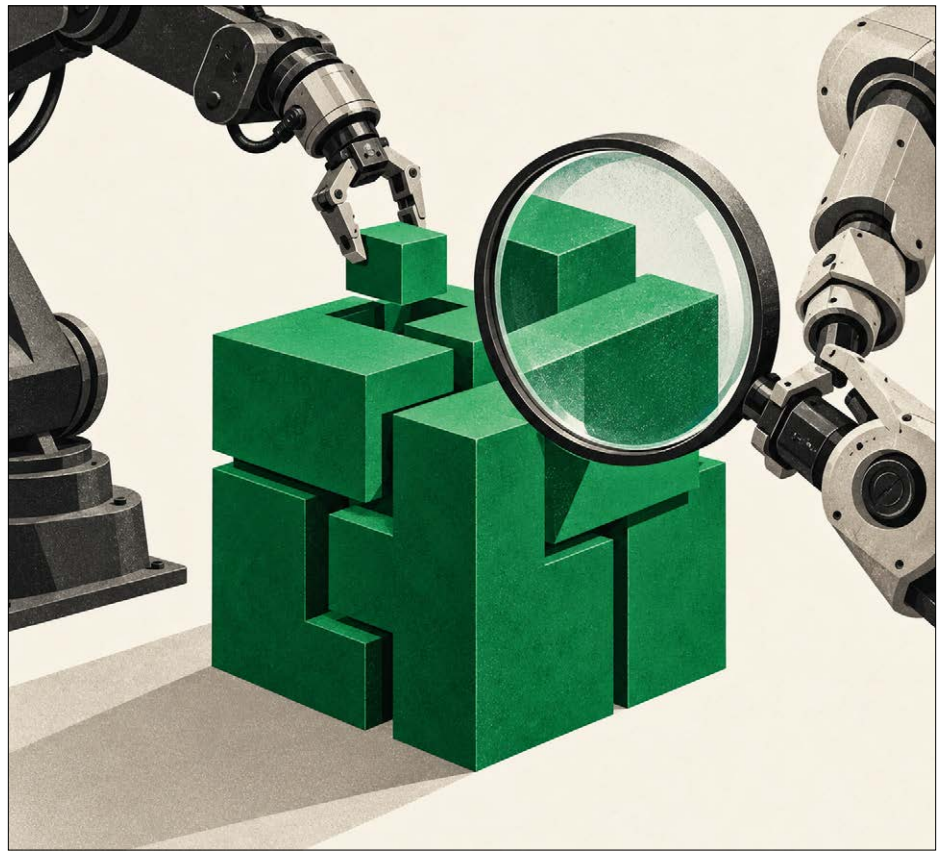
Второе преимущество — многоуровневая защита. OSA/SCA, SAST, DAST, фаззинг и другие практики исследуют приложение на разных уровнях и этапах. Независимые результаты можно перекрёстно валидировать. Сочетание методов даёт более широкое покрытие, чем использование одного ИИ-анализатора в конце процесса.

Третье преимущество — разделение ролей между моделями. Генерация

кода и анализ его безопасности требуют разных режимов работы. Модель, которая создала код, не должна оставаться его единственным проверяющим.

Поэтому конвейер предполагает двухмодельную архитектуру. Одна модель используется для генерации кода, другая — для анализа, классификации и исправления уязвимостей. Например, для этих ролей можно использовать Claude Sonnet и модели OpenAI o-серии. Конкретный выбор моделей вторичен: принципиально разделение функций между разными ИИ-контурами.

Результаты «Солара» также показывают пользу специализации. В задачах триажа и кодфикса специализированные модели показывали более высокую точность, чем публичные универсальные LLM.



## РИСКИ САМОГО ИИ-КОНТУРА

Чем больше задач решает ИИ, чем больше его инструментов появляется в среде разработки, тем больше компонентов самого конвейера приходится защищать и контролировать. Можно определить следующие основные риски:

Первый риск связан с ошибками моделей. Галлюцинации могут привести к пропуску реальной уязвимости или появлению ложного срабатывания. Ошибка возможна и при генерации исправления.

Второй риск — раскрытие чувствительной информации при использовании внешних ИИ-сервисов. В первом полугодии 2026 года ГК «Солар» проанализировала 12 тыс. взаимодействий сотрудников крупных российских организаций с публичными ИИ-сервисами. В 38% случаев взаимодействия содержали конфиденциальную информацию. Среди них 41% пришлось на исходный код и конфигурации. На команды разработки пришлось 43% таких событий.

Третий риск связан с цепочкой поставки. ИИ-ассистент может предложить небезопасный или вредоносный программный компонент. Поэтому внешние зависимости должны проходить отдельную проверку.

Четвёртый риск — полномочия агентов. Агент может работать с репозиторием, инструментами разработки и тестовой средой. В отдельных сценариях он получает доступ и к другим системам. Поэтому агенту нужны ограниченная роль, минимальные права и журналирование действий.

Потребность в такой защите уже фиксируют российские компании. В совместном исследовании Группы компаний Б1, ГК «Солар», Ассоциации ФинТех и HiveTrace защиту интеллектуальных агентов назвали востребованной функцией специализированных средств защиты ИИ 27% респондентов.

Особого контроля требуют агенты, которые управляют средствами тестирования. Такой агент не должен выходить за установленную область проверки. Иначе возможны несанкционированные, в том числе деструктивные действия в других компонентах инфраструктуры.

Наконец, целью атаки может стать сам ИИ-контур. Его компрометация позволяет воздействовать сразу на несколько стадий разработки. Под угрозой оказываются требования, архитектура, код, результаты анализа и исправления. Поэтому безопасность ИИ-контура становится частью безопасности всей цепочки поставки ПО.

## ИИ КАК ЭЛЕМЕНТ СОВРЕМЕННОГО РБПО

В этой модели применение ИИ не сводится к генерации кода или проверке готового приложения. Он участвует в формировании требований, моделировании угроз, проектировании архитектуры, разработке, анализе, классификации и исправлении уязвимостей.

Специализированные агенты при этом не заменяют средства анализа и человека. SCA, SAST, DAST и другие методы дают независимые результаты. Агенты обрабатывают результаты проверок и передают необходимый контекст между этапами. Разные ИИ-контуры разделяют генерацию и проверку. Финальное решение принимает специалист.

По мере распространения ИИ в разработке он становится неотъемлемым элементом современного процесса РБПО.

# КАК ВНЕДРИТЬ ПРОЦЕССЫ БЕЗОПАСНОЙ РАЗРАБОТКИ ПО В МАЛОМ БИЗНЕСЕ

## ПРАКТИЧЕСКИЙ ОПЫТ НЕБОЛЬШОЙ КОМАНДЫ НА ПУТИ К СООТВЕТСТВИЮ ГОСТ Р 56939–2024



**Илья АНТИПОВ**
руководитель группы Центра оценки соответствия и тестирования АО «НПО «Эшелон»



**Нелли МАГАКЕЛОВА**
руководитель группы Центра оценки соответствия и тестирования АО «НПО «Эшелон»

**За разработкой безопасного ПО (РБПО) закрепилась репутация сложного и дорогого направления, доступного только крупным вендорам с отдельными департаментами ИБ. Стандарт описывает двадцать пять процессов, и на первый взгляд их внедрение кажется непосильным для команды из нескольких человек. В реальности, выстроив повторяемый и контролируемый процесс, небольшая организация получает предсказуемый результат на каждом релизе.**

### ВАЖНОСТЬ ВЫСТРОЕННОГО КОНВЕЙЕРА РАЗРАБОТКИ

Значительная часть требований стандарта выполняется не дополнительными средствами защиты, а грамотно настроенным конвейером разработки и дисциплиной команды. Достаточно посмотреть на то, как устроен GitLab, чтобы увидеть, что многие процессы закрываются его штатными возможностями.

К примеру, использование системы контроля версий вместе с тегами релизов отвечает за управление конфигурацией (процесс 5.4). Система issues и баг-трекеры обеспечивают управление недостатками и запросами на изменение (процесс 5.5). Подписанные коммиты, защищённые ветки с настройкой контроля доступа и двухфакторная аутентификация закрывают контроль целостности и доступа к коду (процесс 5.14). Запросы на слияние с принятыми в организации правилами согласования реализуют перекрёстное ревью, то есть экспертизу исходного кода (процесс 5.9). Сборка на изолированных раннерах отвечает требованиям к безопасной системе и среде сборки (процессы 5.12 и 5.13).

Иными словами, заметная доля процессов выполняется уже за счёт того, что команда работает в современной платформе контроля версий. Это принципиально меняет оценку трудозатрат, поскольку выстраивать названные процессы с нуля не нужно, достаточно корректно настроить то, что используется каждый день.

### ДОПОЛНИТЕЛЬНАЯ ЗАЩИТА КОНВЕЙЕРА РАЗРАБОТКИ

Поверх штатных возможностей платформы выстраивается автоматический анализ кода, который превращает обычную разработку в безопасную. Ещё до попадания кода в репозиторий pre-commit-хуки автоматически проверяют каждый коммит на наличие секретов и соблюдение правил кодирования. Далее при каждом изменении конвейер запускает статический и композиционный анализ, поиск секретов, а также модульные и регрессионные тесты. Более тяжеловесные проверки (фаззинг-тестирование, пентест) выполняются в отдельном тестовом контуре по расписанию или перед выпуском версии, в первую очередь для модулей, входящих в поверхность атаки.

Важную роль здесь играют security gates — точки конвейера, в которых код продвигается дальше только при выполнении заданных условий безопасности. Первый барьер блокирует merge request, если статический или композиционный анализ выявил дефекты и уязвимости критического или высокого уровня либо в коде обнаружен секрет. Второй барьер стоит перед выпуском версии и учитывает результаты динамического анализа и тестирования собранного ПО. В GitLab такие барьеры задаются внутренними политиками, которые хранятся в репозитории вместе с кодом, поэтому разработчик видит результат проверки прямо в запросе на слияние.

Результаты всех инструментов передаются в оркестратор дефектов, например DefectDojo, который объединяет повторяющиеся срабатывания, ранжирует их по критичности и автоматически создаёт задачи в трекере, поэтому сотрудник, проводящий разметку, работает уже с единым отчётом из нескольких инструментов. Весь конвейер разворачивается в изолированном контуре без выхода в интернет, а внешние зависимости поступают через контролируемый репозиторий. Общая архитектура такого контура показана на рисунке 1.

Управление кодом в системе контроля версий обеспечивает выполнение процессов 5.4, 5.5, 5.9 и 5.14.

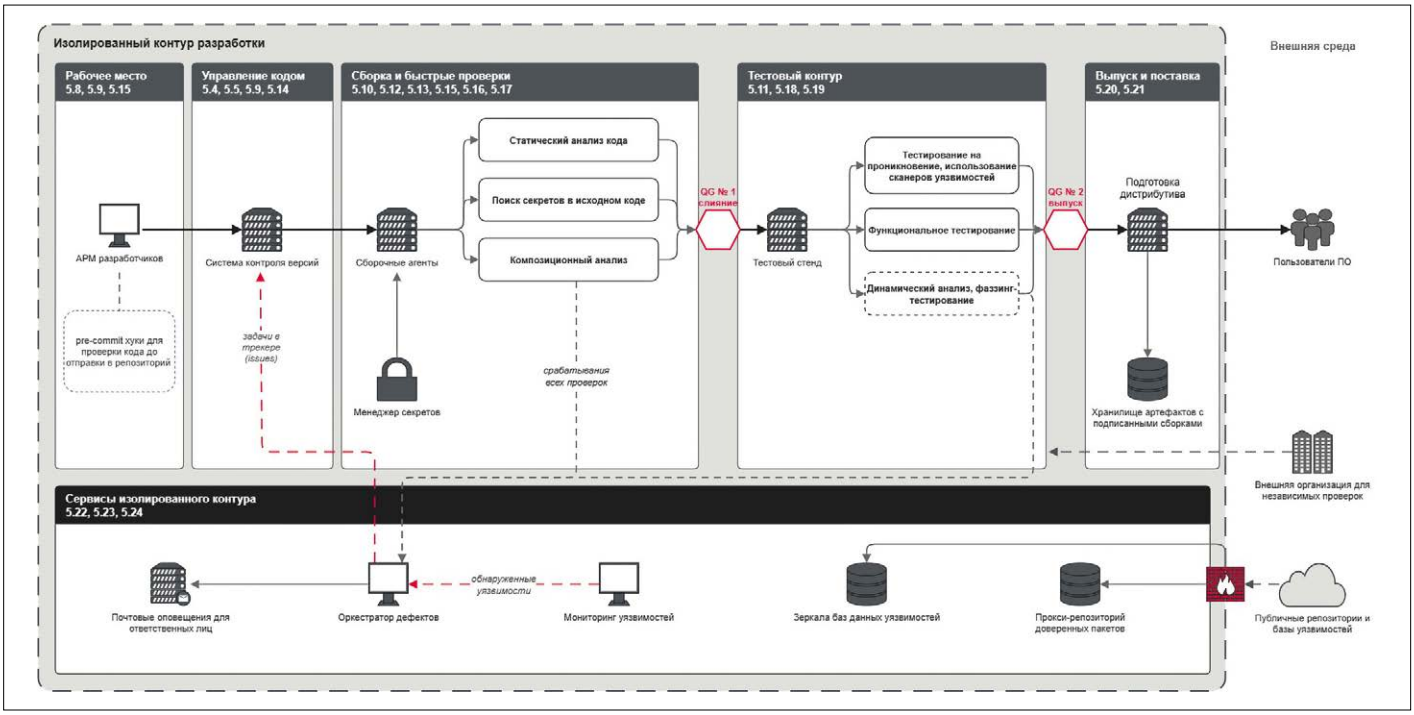



**Рисунок 1.** РБПО-конвейер и закрываемые им процессы ГОСТ Р 56939–2024

На участке сборки и быстрых проверок реализуются процессы 5.10, 5.12, 5.13 и 5.15–5.17. Тестовый контур закрывает процессы 5.11, 5.18, 5.19, а выпуск и поставка — процессы 5.20 и 5.21. Сервисы изолированного контура поддерживают сопровождение продукта, реагирование на уязвимости и их поиск при эксплуатации (5.22–5.24). Оставшиеся процессы 5.1–5.3, 5.6, 5.7 и 5.25 выполняются вне конвейера. В совокупности конвейер и организационные практики охватывают все двадцать пять процессов стандарта.

## КОМАНДА И РАСПРЕДЕЛЕНИЕ РОЛЕЙ

Процессы в небольшой команде распределяются между сотрудниками, которые уже выполняют смежные задачи.

DevOps-инженер отвечает за автоматизацию, встраивает инструменты анализа в конвейер, настраивает сборку, управление секретами и подписание релизов. В небольших командах роль security champion нередко поручают одному из разработчиков, однако такой подход нельзя назвать удачным. Разработчик в первую очередь должен выполнять свои прямые обязанности, а объективно оценить безопасность собственного кода ему сложно. Поэтому на эту роль целесообразно выделить отдельного специалиста, который отвечает за все процессы РБПО, включая проведение узкопрофильных видов тестирования, таких как фаззинг и пентест. Разработчики, в свою очередь, отвечают за формирование правил кодирования и устранение всех найденных дефектов и уязвимостей. Тестировщик обеспечивает функциональное и нефункциональное тестирование. Аналитик вместе с архитектором формирует требования безопасности к ПО, обеспечивает пересмотр архитектуры ПО с точки зрения безопасности и отвечает за моделирование угроз безопасности ПО совместно со специалистом по безопасности. Руководитель команды отвечает за общую организацию всех процессов, в том числе планирование, обучение сотрудников, сопровождение и вывод продукта из эксплуатации. Также независимое тестирование на проникновение и поиск недекларированных возможностей (НДВ) может выполнять внешняя организация. Периодическое привлечение независимых специалистов полезно, поскольку команда, которая долго работает с собственным кодом, со временем перестаёт замечать в нём слабые места, а взгляд со стороны выявляет то, что пропускают внутренние проверки.

Документальная часть не менее важна, так как без регламентов не определены зоны ответственности и не зафиксирован порядок выполнения процессов. При этом регламенты должны описывать реально применяемую практику. Свидетельства, то есть документы и артефакты, подтверждающие выполнение требований, перечислены для каждого процесса в пунктах 5.X.3 «Артефакты реализации требований». Значительную часть из них, включая отчёты анализаторов, Software Bill of Materials (SBOM), журналы сборки и решения security gates, конвейер формирует автоматически.

## ВЫВОД

Внедрение РБПО в небольшой команде не требует ни отдельного подразделения по безопасности, ни значительных трудозатрат. Достаточно грамотно настроенного Continuous Integration / Continuous Delivery or Deployment (CI/CD) пайплайна со встроенными мерами защиты и инструментария, понятных регламентов и нескольких организационных практик. Выстроенный таким образом процесс сохраняет привычный ритм работы команды и одновременно обеспечивает соответствие стандарту.



# SAST 2026

## УЗКОЕ МЕСТО БЕЗОПАСНОЙ РАЗРАБОТКИ СМЕСТИЛОСЬ ОТ ПОИСКА УЯЗВИМОСТЕЙ К ИХ ИСПРАВЛЕНИЮ



**Александр ЗАВОДОВСКИЙ**
руководитель продукта AppSec.Wave, AppSec Solutions (ГК Swordfish Security)

**Последние годы рынок статического анализа соревновался в одном: кто точнее найдёт дефект и меньше «нашумит» ложными срабатываниями. Эту гонку во многом можно считать выигранной: зрелые анализаторы восстанавливают потоки данных, а ИИ-триаж снимает с инженера заметную часть разметки. Но если проследить путь находки до конца, выясняется, что код от этого быстрее безопасным не становится.**

### РАЗБОР УСКОРИЛСЯ, ИСПРАВЛЕНИЕ ЗАМЕДЛИЛОСЬ

Мы сравнили медианное время жизни «находок» SAST в 2025 и 2026 годах на двух отрезках: от обнаружения до вердикта и от вердикта до исправления. Первый сократился почти вдвое на всех уровнях критичности. Второй, напротив, вырос. Для критичных находок медиана увеличилась примерно со 100 до 150 дней, для находок высокого уровня — вдвое. Узкое место не исчезло, оно переехало за пределы сканера, и подтверждённая уязвимость теперь дольше ждёт исправления, чем раньше ждала подтверждения.

### ПЕРВЫЕ РЕЗУЛЬТАТЫ ИИ-АВТОФИКСА

Логичный ответ — передать исправление искусственному интеллекту (ИИ), и первые данные выглядят обнадёживающе. По опубликованным отраслевым замерам, ИИ-подсказка исправления сокращает время закрытия свежей находки примерно втрое, а для отдельных типов уязвимостей — более чем в десять раз.



Неудивительно, что ИИ-триаж и автоисправление сегодня появляются в дорожных картах большинства вендоров статического анализа, в том числе российских.

Но эти цифры, как правило, описывают свежий код в pull request, где дефект и исправление обычно находятся рядом. Долг безопасности устроен иначе. Это старый код, который давно никто не открывал, и дефекты, разнесённые по нескольким модулям.

## ПОЧЕМУ ОДНОЙ СТРОКОЙ НЕ ОБОЙТИСЬ

Уязвимость редко живёт в одной строке. Обычно это цепочка: данные от пользователя проходят через несколько частей программы, прежде чем попасть туда, где они становятся опасны. Анализатор указывает на последнее звено, где проблема проявилась. А исправлять нужно там, где данные входят в систему, и по пути их движения. В системах на 1С, которые десятилетиями росли через доработки и нередко обрабатывают персональные данные, в одном модуле формы такая цепочка может насчитывать 25 промежуточных шагов, и ни один из них по отдельности не выглядит уязвимостью.

## АВТОФИКС, КОТОРЫЙ ВИДИТ ВСЮ ЦЕПОЧКУ

ИИ-исправление должно опираться на анализ потоков данных, то есть на полную карту их движения по программе, а не на одну строку с предупреждением.

Работа делится на две части. Анализатор отвечает на вопрос «где»: прослеживает путь данных через все файлы и видит, какие ещё участки программы зависят от тех же функций.

ИИ отвечает на вопрос «как»: составляет для разработчика понятный план — какие изменения внести, где и зачем нужно каждое из них. После правки анализатор проверяет код снова, чтобы определить, закрыт ли путь и не появилось ли новых проблем. Если одна уязвимая функция затрагивает десятки мест, её исправление закрывает их все. Разработчик получает не диагноз «здесь уязвимость», а готовый план работ.

Поэтому мы начали с глубины и точности трасс. На примере 1С: в статическом анализе поддержка этого языка сводится к поиску по шаблонам в тексте конфигурации, а в решении AppSec.Wave реализован полноценный анализ потоков данных с деревом вызовов в модуле и поиском обхода ограничений доступа на уровне записей (RLS), небезопасных внешних обработок и прямых запросов в обход прав. Автоисправление, построенное на таких трассах,— именно то, над чем сейчас работает команда AppSec.Wave.

## ЧТО СПРОСИТЬ У ВЕНДОРА ПРИ ВЫБОРЕ SAST-АНАЛИЗАТОРА

При выборе инструмента стоит выяснить, охватывает ли предложенное автоисправление весь путь данных или только строку срабатывания; учитывает ли оно другие вызовы изменяемой функции; проверяется ли результат повторным анализом.

# CODESCORING.SAVE

## КАК ВЫГЛЯДИТ ХРАНИЛИЩЕ АРТЕФАКТОВ, СПРОЕКТИРОВАННОЕ С УЧЁТОМ СОВРЕМЕННЫХ УГРОЗ



**Дмитрий КЛЮЧНИКОВ**
руководитель разработки CodeScoring.Save

**Рынок управления артефактами долго казался устоявшимся: на слуху универсальные Nexus Repository и JFrog Artifactory, Harbor как OCI-совместимый реестр, встроенные реестры платформ разработки (IDP), а для ускоренной раздачи образов есть новаторские P2P-решения вроде Dragonfly. Казалось, достаточно выбрать инструмент под свой стек, требования к доступности и стоимость владения.**

Но уход зарубежных вендоров и жёсткие ограничения бесплатных версий разрушили привычный сценарий: коммерческие лицензии недоступны, а open-source редакции отрезают кластеризацию, отказоустойчивость и безопасность. На практике выбор сузился, а за каждым доступным решением проявились архитектурные компромиссы: ни одно из них изначально не проектировалось под современные требования к защите цепочки поставки ПО.

В 2026 году компания CodeScoring выпустила собственное хранилище артефактов CodeScoring.Save. Почему мы решили написать систему с нуля и чем она отличается от привычных менеджеров репозиториев — в материале Дмитрия Ключникова, руководителя разработки CodeScoring.Save.

### ЭВОЛЮЦИЯ АРТЕФАКТОВ И НОВЫЙ КОНТЕКСТ УГРОЗ

Артефактом сегодня называют не только библиотеку или пакет. Это контейнерные образы, Helm-чарты, модели машинного обучения (ML-модели), скомпилированные бинарники и другие объекты, которые создаются или используются в процессе разработки и далее движутся к развёртыванию. У каждого такого объекта есть жизненный цикл: его нужно разместить, версионировать, выдать потребителю и, что в контексте новых угроз становится всё важнее, проверить на безопасность.

Раньше вопрос выбора хранилища сводился к инженерным параметрам: пропускной способности, набору поддерживаемых форматов, качеству API и удобству проксирования. Сегодня на первый план выходит другой критерий: можно ли доверять тому, что лежит в репозитории.

Согласно исследованию Cloudsmith, уже 56% команд называют защиту цепочки поставки главной задачей систем управления артефактами. Основания для этого весомые. За 2025 год аналитики CodeScoring зафиксировали 457 тысяч вредоносных пакетов в открытых источниках — рост в 11 раз относительно предыдущего года. Параллельно база открытых компонентов пополнилась 14 тысячами новых уязвимостей.

Новый вектор атак открыло массовое внедрение AI-ассистентов генерации кода. Исследование USENIX показало, что даже у пятёрки ведущих моделей от 4,6% до 6,1% рекомендуемых пакетов не существуют в реальных репозиториях (а у менее мощных и бесплатных решений этот разброс существенно выше). Поскольку модели повторяют одни и те же вымышленные имена, атакующие используют эту предсказуемость для техники slopsquatting: регистрируют фантомные имена во внешних реестрах и внедряют вредоносный код, ожидая запросов от невнимательных разработчиков. В этой схеме внутреннее хранилище становится рубежом обороны, отсекающим сомнительные зависимости на входе в контур согласно заданным политикам.

### ЧЕТЫРЕ АРХИТЕКТУРНЫХ ПОДХОДА: ВОЗМОЖНОСТИ И КОМПРОМИССЫ

Каждое хранилище артефактов можно не только описать как набор функций, но и увидеть за ним определённую философию.

- Универсальные монолиты, Nexus Repository Manager и JFrog Artifactory, во многом сформировали привычную модель «менеджера репозиториев на все случаи жизни» с десятками экосистем под одной оболочкой, зрелыми механизмами репликации и обширной базой внедрений. Обратная сторона зрелости — вес и инерция архитектуры 2010-х годов. Высокая доступность, распределённая кластеризация и продвинутые security-функции исторически заложены

**За 2025 год аналитики CodeScoring зафиксировали 457 тысяч вредоносных пакетов в открытых источниках — рост в 11 раз относительно предыдущего года. Параллельно база открытых компонентов пополнилась 14 тысячами новых уязвимостей**



только в платных редакциях, которые сегодня недоступны на российском рынке. При этом в open-source версиях отказоустойчивость ограничена или отсутствует, а обновления на терабайтных объёмах несут риск поломки схемы данных и плагинов, поэтому их годами откладывают ценой накопления уязвимостей.

◆ Узкоспециализированные решения реализуют противоположную стратегию — «чем проще, тем лучше». Harbor и Zot, например, ориентированы строго на OCI-совместимые артефакты и предлагают баланс между функциональностью и простотой: kube-native архитектуру, встроенные сканеры и работу с цифровыми подписями. Но есть и жёсткое ограничение: если в стеке есть Maven, npm или PyPI или любой другой формат не из Open Container Initiative (OCI)-мира, рядом придётся ставить второе хранилище со своими политиками безопасности, циклом обновлений и интеграцией с Continuous Integration / Continuous Delivery or Deployment (CI/CD).

◆ Третий подход, «хранилище как часть платформы», реализуют решения класса Internal Developer Platform (IDP), например, GitLab или Azure DevOps Server. Порог входа минимальный: пользователи получают единый интерфейс, общие права, нативную связку с CI. Но интеграции с внешними ИБ-инструментами при этом ограничены, поддержка пакетных менеджеров нередко реализована по остаточному принципу, а само хранилище невозможно масштабировать независимо от серверов основной платформы.

◆ Наконец, есть решения-надстройки дистрибуции «поверх хранилищ». Dragonfly и подобные проекты меняют саму логику доставки: образы распространяются между узлами по P2P-схеме, снижая нагрузку на центральный реестр до 90%. Но это не самостоятельное хранилище, а дополнительный сервисный слой, который требует отдельной поддержки и наиболее эффективен на масштабах

сотен узлов. Для большинства компаний это скорее интересный пример, чем решение для повседневной практики.

У каждого из существующих решений есть своя ниша, но рынок в целом упёрся в три фундаментальных пробела:

- **Первый: архитектурный**. Универсальные монолиты несут балласт 15-летнего легаси, платформенные реестры привязывают пользователя к одной экосистеме, а узкоспециализированные решения ограничивают работу только форматом OCI.
- **Второй: безопасность реализована как отдельный слой**. В традиционном стеке она существует параллельно с хранением: Nexus подключает IQ Server, JFrog — Xray, Harbor использует сторонние сканеры. Хранилище и безопасность не объединены на уровне политик: проверки часто происходят асинхронно постфактум, а блокировка на входе реализуется через внешние плагины или вебхуки. В итоге инструментам безопасности не хватает контекста из хранилища (ограничения сторонних API), а сам реестр не имеет сквозного понимания происхождения компонентов и структуры релиза.
- **Третий: российская нормативная специфика**. Ни одно зарубежное решение по понятным причинам не учитывает требования отечественных стандартов безопасной разработки, включая ГОСТ Р 56939–2024 и связанные с ним практики разработки безопасного ПО (РБПО), а также не ориентировано на контролируемую работу с российскими дистрибутивами и экосистемами.

## ОТЛИЧИЯ CODESCORING.SAVE

CodeScoring.Save — собственная разработка CodeScoring. В отличие от традиционных Java-монолитов, требовательных к ресурсам Java Virtual Machine (JVM) и требующих много памяти в любой момент работы (даже в простое), решение написано на Go: статические бинарники обеспечивают минимальное потребление памяти, быстрый старт и запускаются в любой контейнерной среде — от публичных облаков до корпоративных платформ. Поставка идёт Helm-чартом, а базовое развёртывание занимает минуты.

Разработка хранилища с чистого листа позволила не тянуть за собой устаревшие архитектурные решения и сразу заложить то, что раньше приходилось достраивать.

- **Разделение compute и storage**. Логика обработки полностью отделена от уровня данных. Stateless-сервисы берут на себя бизнес-логику и обработку протоколов пакетных менеджеров, метаданные хранятся во внешней PostgreSQL или встроенной SQLite, а бинарные файлы — в корпоративном S3-совместимом объектном хранилище либо на файловой системе. Продукт органично переиспользует уже имеющуюся инфраструктуру заказчика.
- **Масштабирование через инфраструктуру**. Задачи масштабирования и балансировки нагрузки вынесены в Kubernetes. За счёт stateless-архитектуры вычислительные сервисы масштабируются горизонтально штатными средствами оркестратора под текущую нагрузку CI/CD без риска рассинхронизации данных.
- **Отказоустойчивость**. Состояние вынесено наружу, поэтому, если поды перезапустились или переехали на другой узел кластера, данные остаются на месте, а при отказоустойчивой конфигурации система может продолжить работу без заметного перерыва.
- **Безопасность внутри, а не «сбоку»**. Безопасность в CodeScoring.Save — не отдельный сервис или лицензия, а слой внутри логики обработки запросов. Уже сейчас работают ролевая модель доступа, сервисные robot-аккаунты для пайплайнов, аудит действий пользователей, интеграция с CodeScoring.OSA и базой уязвимостей платформы. В планах — более подробная информация об уязвимостях, расширение связки с инструментами анализа безопасности ПО, включая антивирус, а также более углублённая работа с provenance: происхождением и маршрутом движения компонентов.
- **Управление через API**. Полноценное управление системой через интерфейсы программирования. В CodeScoring.Save REST API и Swagger позволяют описывать конфигурацию как код, бесшовно встраивать хранилище в CI/CD и строить собственные автоматизации.

Уже сегодня CodeScoring.Save поддерживает популярные экосистемы, включая Maven, npm, PyPI, NuGet, Go, Docker/OCI, Deb (APT), RPM (YUM/DNF), Raw, Composer (PHP), Alpine (APK) и Astra — с поддержкой hosted- и proxy-режимов. Для быстрого переезда предусмотрен инструмент автоматической миграции с открытых репозиториев.

Ближайшие направления развития продукта — подключение новых пакетных менеджеров, углубление проверок безопасности артефактов и развитие provenance. Отдельный фокус сделан на безопасность AI-пайплайнов: уже тестируется хранение и раздача ML-моделей (Hugging Face), что позволит безопасно размещать и проксировать веса моделей внутри корпоративного периметра, защищая контур от непроверенных внешних AI-компонентов.

Отдельный стратегический трек — Release Lifecycle Management: переход от контроля единичных файлов к работе с релизом как целостной сущностью (включая связанные образы, конфигурации и Software Bill of Materials (SBOM)). Именно такая логика заложена в концепцию РБПО и наиболее востребована российскими enterprise-заказчиками.

CodeScoring.Save последовательно закрывает пробелы существующих решений:

- архитектурный за счёт разработки с нуля и гибкой структуры,
- пробел в безопасности за счёт встроенного контроля политик при обращении к реестру,
- и нормативный за счёт ориентации на отечественные стандарты безопасной разработки.

CodeScoring.Save подойдёт командам, которым необходимо быстрое, предсказуемое и отказоустойчивое хранилище — и особенно тем, кто стремится сделать свой репозиторий контролируемой точкой входа для безопасных компонентов.

# БЕЗОПАСНОСТЬ ЦЕПОЧКИ ПОСТАВКИ ПО: ОТ ИСХОДНОГО КОДА ДО РЕЗУЛЬТАТА СБОРКИ

## КАК ВЫСТРОИТЬ ЗАЩИЩЁННУЮ СРЕДУ РАЗРАБОТКИ С УЧЁТОМ ПРОЕКТА ГОСТ «СРЕДА РАЗРАБОТКИ БЕЗОПАСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»



**Евгений КУЛИКОВ**
руководитель группы обеспечения безопасности приложений, Swordfish Security

**Безопасность программного продукта сегодня зависит от всего пути его создания, включая сторонние компоненты, сборку и поставку. Проект ГОСТ «Среда разработки безопасного программного обеспечения» отражает этот подход, и готовиться к нему стоит уже сейчас.**

Внешние библиотеки, средства сборки, контейнерные образы и хранилища влияют на конечный результат наравне с собственным кодом разработчиков. Поэтому одной проверки исходного кода уже недостаточно, и организации важно контролировать каждое звено цепочки поставки. В статье разберём, что предлагает проект ГОСТ, как пошагово подготовиться к его требованиям и на какие международные практики при этом опираться.

### ЦЕПОЧКА ПОСТАВКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Безопасность цепочки поставки программного обеспечения (ПО) отвечает на вопрос, может ли организация доказать, из чего, где и каким образом был создан продукт, поставленный заказчику или запущенный в эксплуатацию. Цепочку удобно рассматривать как последовательность этапов, от внешних компонентов до поставки (рис. 1).

На каждом этапе возникают свои риски. Внешняя библиотека может содержать уязвимость или вредоносный код, исходный код может быть изменён неавторизованным пользователем, система сборки может быть скомпрометирована без изменения исходников, а готовый результат может быть подменён перед поставкой. Поэтому одного анализа исходного кода недостаточно, необходимо контролировать всю последовательность создания продукта.

### ЧТО ПРЕДЛАГАЕТ ГОТОВЯЩИЙСЯ ГОСТ

Проект ГОСТ «Защита информации. Разработка безопасного программного обеспечения. Среда разработки безопасного программного обеспечения. Общие требования» охватывает всю среду разработки целиком. Документ пока находится в статусе проекта первой редакции, поэтому его следует воспринимать как ориентир для подготовки и развития процессов, а не как действующий обязательный стандарт.

Логика проекта строится вокруг контроля всего жизненного пути программного продукта внутри среды разработки. Организации необходимо понимать, какие системы и инструменты участвуют в разработке, какие из них являются критичными и какие угрозы для них актуальны. Отдельные требования посвящены использованию стороннего программного обеспечения, защите исходного кода, управлению сборкой, изоляции и контролю сборочной среды, а также обеспечению целостности, хранения и прослеживаемости результатов. Такой подход смещает фокус с отдельных проверок ис-




**Рисунок 1.** Цепочка поставки программного обеспечения

ходного кода на защиту всего процесса создания ПО — от подключения внешних компонентов до формирования версии, готовой к поставке.

## НА КАКИЕ ПОДХОДЫ МОЖНО ОПЕРЕТЬСЯ

Практики, близкие к требованиям проекта ГОСТ, уже описаны в открытых международных методиках S2C2F, OWASP SCVS и SLSA. Удобно воспринимать их как взаимодополняющие подходы.

S2C2F (Secure Supply Chain Consumption Framework, Microsoft) отвечает за управление внешними компонентами и определяет, откуда их можно получать, как разрешать, проверять, обновлять и блокировать. OWASP SCVS (Software Component Verification Standard) задаёт перечень свойств компонентов и окружения, которые должны быть проверяемыми, в том числе состав, источник, версию, контрольную сумму, зависимости и параметры среды. SLSA (Supply-chain Levels for Software Artifacts, OpenSSF) позволяет выстроить проверяемую связь между исходным кодом, процессом сборки и полученным результатом.

Проще говоря, с помощью S2C2F мы контролируем то, что используем, с помощью SCVS определяем, что проверяем, а с помощью SLSA доказываем, как получен результат.

## ПОШАГОВАЯ МОДЕЛЬ ПОДГОТОВКИ

**Шаг 1. Провести инвентаризацию среды разработки.** Первым делом нужно определить, что участвует в создании продукта. Для этого составляется перечень систем и инструментов. В него входят хранилище исходного кода, средства сборки, хранилище результатов, системы управления секретами, тестовые среды, контейнерная и виртуальная инфраструктура. Для каждой системы фиксируются назначение, владелец, версия, сетевые связи и зона ответственности. В итоге мы получаем карту среды разработки, которая показывает критичные точки и ответственных. Этот шаг соответствует разделу 5 проекта ГОСТ и практикам SCVS по инвентаризации среды.

**Шаг 2. Взять под контроль внешнее ПО.** На этом этапе важно исключить прямую и бесконтрольную загрузку библиотек и компонентов из внешних источников. Стороннее ПО должно поступать только через контролируемый репозиторий. До размещения оцениваются источник, целостность, подпись поставщика при наличии, известные уязвимости и признаки вредоносного содержимого. Таким образом, компонент попадает в разработку и сборку только после проверки и размещения в контролируемом хранилище (рис. 2). Выстраивая этот процесс, можно опереться на S2C2F, где описан порядок получения и допуска внешних компонентов, и на OWASP SCVS, который определяет состав необходимых проверок.

Проверка должна быть постоянной, так как после допуска компонента могут появиться новые уязвимости. Поэтому нужны периодический анализ, контролируемое обновление и политики безопасности «Allow / Review / Deny». Важно также быстро определять продукты, где используется проблемная версия компонентов.

**Шаг 3. Защитить исходный код.** Когда внешние компоненты под контролем, следующей точкой риска становится исходный код. Важно, чтобы любое изменение в нём можно было отследить до конкретного автора. Поэтому код хранится в системе контроля версий, где каждый пользователь работает под персональной учётной записью, права выдаются по принципу минимальных привилегий, а все действия регистрируются. Основные ветки защищены, и правки попадают в релиз только после ревью. Для каждой поставляемой версии сохраняются код и сведения для контроля его целостности. Требования к этому этапу сформулированы в разделе 7 проекта ГОСТ, а практические рекомендации можно найти в SLSA.

**Шаг 4. Сделать сборку управляемой.** Сборка должна выполняться из зафиксированной версии исходного кода в контролируемой среде и без произвольной загрузки компонентов из неизвестных внешних источников. Для каждого результата сохраняются сведения о его происхождении, в том числе об использованном коде, системе сборки и полученном артефакте, а обращения во внешнюю сеть и состав зависимостей фиксируются и контролируются. Ориентиром здесь служит SLSA, где описаны уровни зрелости сборочного процесса. На более высоких уровнях сборка выполняется в изолированной доверенной среде, а сведения о происхождении формируются автоматически и защищены от изменения.

**Шаг 5. Защитить сборочную среду.** Даже управляемая сборка останется уязвимой, если ненадёж-

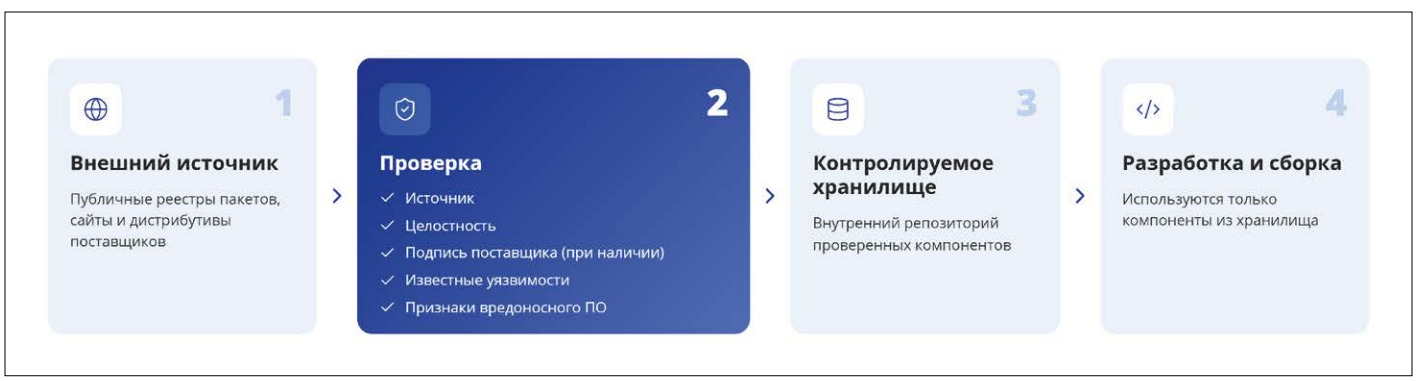



**Рисунок 2.** Получение компонентов через контролируемый репозиторий

на сама сборочная среда. Поэтому она формируется из заранее определённых доверенных компонентов. Инструменты сборки, образы, библиотеки, конфигурации и зависимости поступают только из контролируемого репозитория, используются зафиксированные версии и контрольные суммы, а сетевой доступ к внешним репозиториям и другим неразрешённым источникам ограничивается. Требования к сборочной среде содержатся в разделах 8–9 проекта ГОСТ, а при их реализации можно опереться на SCVS и требования SLSA к сборке.

**Шаг 6. Защитить результат сборки.** После сборки важно убедиться, что в поставку попадает именно тот результат, который прошёл все проверки. Для этого фиксируются его версия и контрольная сумма, а также сохраняются сведения о составе, результатах проверок и самой сборке. Эти данные позволяют связать готовый файл, пакет или контейнерный образ с конкретной сборкой и версией исходного кода. Проект ГОСТ также относит к результатам сборки журналы, конфигурацию и файлы контроля целостности. Подпись подтверждает подлинность результата и позволяет убедиться, что после сборки он не был изменён.

**Шаг 7. Сохранить непрерывность доверия до поставки.** Для сохранения доверия к результату от сборки до поставки можно использовать несколько взаимодополняющих подходов. Проект ГОСТ задаёт требования к хранению, контролю доступа, целостности и проверке эталонных версий перед поставкой. Связать результат с конкретной сборкой и исходным кодом позволяет SLSA, а определить, какие сведения о составе и происхождении должны сохраняться и проверяться, помогает OWASP SCVS.

**Шаг 8. Собирать доказательства.** Для каждой сборки необходимо сохранять данные, позволяющие восстановить ход процесса и подтвердить происхождение результата: время, версию, конфигурацию, основные этапы, ошибки, действия пользователей, отчёты о сканированиях и другие значимые сведения. В логике SLSA эти данные обеспечивают прослеживаемую связь между исходным кодом, процессом сборки и полученным результатом.

## КАК ВСТРОИТЬ ТРЕБОВАНИЯ В ОБЫЧНЫЙ ПРОЦЕСС РАЗРАБОТКИ

Контроль можно внедрить непосредственно в процесс разработки через несколько последовательных точек принятия решения. До допуска внешнего компонента проверяются его источник, версия, целостность и риски, выявленные в ходе проверок. При изменении исходного кода оцениваются новые зависимости и результаты проверок. На этапе сборки подтверждается использование зафиксированной версии исходного кода, разрешённых зависимостей и сборочной среды. Перед поставкой проверяются целостность и подпись результата, а также его соответствие сборке, которая прошла все установленные проверки и допущена к поставке.

Такой подход позволяет встроить требования безопасности в обычный процесс разработки. Результаты проверок в каждой точке становятся основанием для решения о том, продвигать результат дальше по цепочке поставки или заблокировать его (рис. 3).

## ИТОГ

Проект ГОСТ рассматривает безопасность разработки как единый процесс, который включает использование сторонних компонентов, работу с исходным кодом, сборку, защиту сборочной среды и контроль готового результата. Подготовку к этим требованиям целесообразно выстраивать поэтапно. Сначала стоит определить состав среды разработки, используемые системы и ключевые взаимосвязи, затем взять под контроль внешние компоненты, защитить исходный код, сделать процесс сборки управляемым, обеспечить целостность результата и сохранять подтверждающие сведения.

S2C2F при этом помогает организовать работу со сторонними компонентами, SCVS позволяет определить, что именно необходимо проверять, а SLSA служит для подтверждения происхождения результата сборки. В итоге формируется последовательный и контролируемый процесс создания и поставки программного продукта.

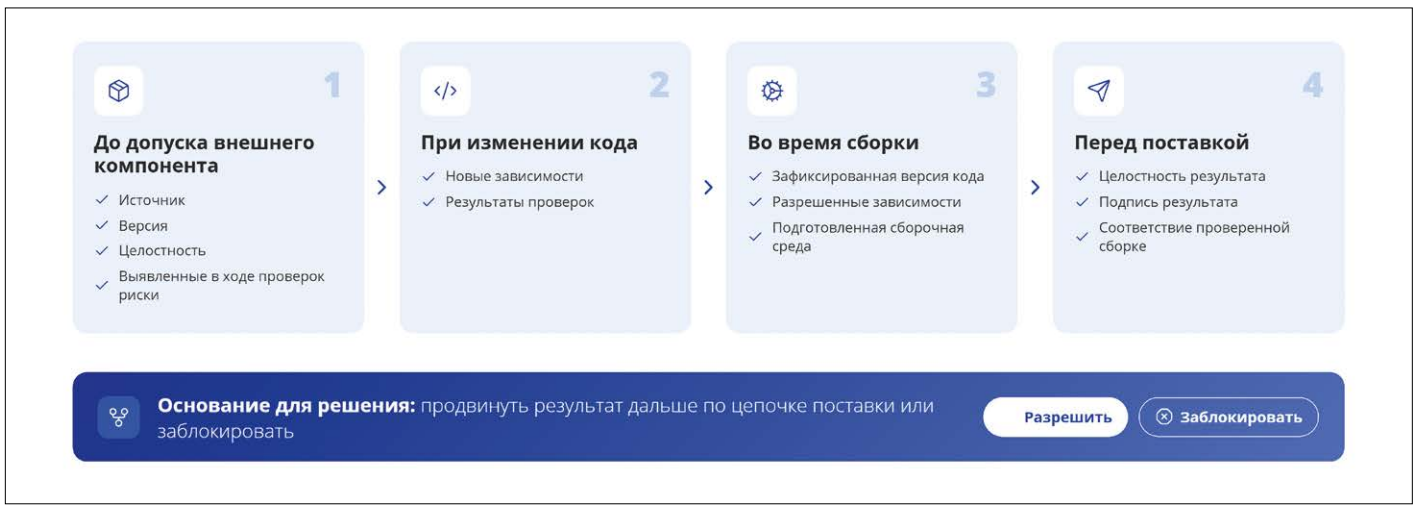



**Рисунок 3.** Точки принятия решений в процессе разработки

# «НЕ ВМЕСТО ЭКСПЕРТА, А РАДИ НЕГО»

## О РОЛИ ИИ В ЛАБОРАТОРИИ




**Артём БУРКЭ**
ведущий эксперт по ИБ Центра исследований ВУ и НДВ НТЦ «Фобос-НТ»



**Андрей КУЗНЕЦОВ**
директор департамента исследований по выявлению уязвимостей и недекларируемых возможностей НТЦ «Фобос-НТ»



**Евгения РЕМЕЗОВА**
начальник отдела технической экспертизы НТЦ «Фобос-НТ»


**НТЦ «Фобос-НТ» — 20 лет. За эти годы на первый план выходили разные подходы к проверке и подготовке материалов: от классических «ручных» (при помощи карандаша и ластика) до разработки систем управления и автоматизации процессов. Сегодня ключевым вектором развития становится интеграция искусственного интеллекта (ИИ) в работу испытательной лаборатории и создание продукта, позволяющего перенести часть накопленной нами за годы экспертизы в ПО, которое, как мы надеемся, поможет нашим партнёрам быстрее проходить сертификационные испытания.**

### ИИ И ДОКУМЕНТАЦИЯ

Для работы лаборатории с заявителем в рамках процесса сертификации мы разработали шаблоны, отвечающие требованиям нормативной документации, отраслевым стандартам и т.д., в которых заявителю оставалось внести свои данные, чтобы сформировать сертификационный пакет документации. Но даже после этого заявители продолжают допускать одни и те же ошибки, которые нам приходится из раза в раз находить и устранять, тратя на это большое количество времени. Помощником в этом стал ИИ. Технические писатели используют его при проверке документации заявителя, анализе исходных данных и подготовке отчётных материалов. На практике это выглядит так: модель анализирует комплект документов, выявляя типичные несоответствия требованиям нормативной документации (приказам и требованиям ФСТЭК России, ГОСТ и т.д.), расхождения информации в документации (например, в контрольных суммах), несоответствия в описании функций безопасности в документации, противоречия между документами, отсутствие прослеживаемости (нет уникального идентификатора, децимального номера, информации об инструменте, который применяется при том или ином испытании и т.д.). После анализа ИИ формирует список замечаний с привязкой к конкретным пунктам нормативных документов и требований, а в документах оставляет комментарии со ссылками на все документы, где ошибка может повториться. Также ИИ применяется при проверке протоколов испытаний: он сопоставляет заявленные цели испытаний с фактическими результатами, а также частично верифицирует применяемые методики тестирования.

Однако внедрение ИИ требует внимания к ряду моментов. Во-первых, результаты работы модели должны обязательно верифицироваться экспертом. ИИ не заменяет инженера или технического писателя, а является лишь инструментом. Во-вторых, важно контролировать качество работы ИИ, чтобы избегать возможных галлюцинаций и «фантазий», когда модель выдумывает несуществующие приказы, нормативные документы или элементы архитектуры. В-третьих, нужно учитывать риски, связанные с обработкой конфиденциальной информации. В нашей лаборатории организована работа локальной модели, не имеющей доступа к сетям общего пользования.

Применение ИИ в работе технических писателей позволило освободить их от рутинных и трудоёмких задач, а также сократить количество ошибок, что дало возможность уделять время более важным, срочным и ответственным задачам.

### ИИ И ППК

Следующим процессом, в который мы внедрили ИИ, стала проверка перечня программных компонентов (ППК). Работа в нём разделена между детерминированными инструментами и моделью, и это разделение принципиально.

Основной инструмент здесь sbom-checker, он проверяет корректность формата ППК, соответствие полей требованиям Методики выявления уязвимостей (ВУ) и недекларированных возможностей (НДВ)[1], формат purl, корректность ссылок на репозитории и версии-заглушки вроде UNKNOWN и 0.0.0. Однако на больших файлах его вывод сложно воспринимается человеком. Наш рекордный

[1] https://cs.groteck.com/IB_4_2026/68/#zoom=z.

ППК был размером 148 МБ и содержал больше трёх миллионов строк. Файл с ошибками тоже впечатлял объёмом и количеством строк. Поэтому поверх sbom-checker используется утилита sbom-report, которая сворачивает тысячи однотипных находок в классы, и уже её вывод мы передаём на анализ ИИ.

ИИ объясняет, почему возникла ошибка, и готовит список замечаний, в котором наглядно указывается, что шесть тысяч предупреждений по purl сводятся к отсутствию namespace у rpm-пакетов. А часть ошибок ИИ исправляет автоматически: например, тип centos-stream, которого нет в спецификации purl, превращается в правильную запись pkg:rpm/centos/...?distro=centos-stream-9. Отдельно модель оценивает правдоподобие разметки модулей, находящихся на поверхности атаки и реализующих функции безопасности.

Одной из самых интересных возможностей нашего решения является сверка ППК с перечнем исполняемых файлов развёрнутого дистрибутива. ИИ сопоставляет имена файлов из предоставленного перечня с содержанием пакетов, которые он индексирует из репозиториев Sisyphus (Альт 8 СП), Debian, Ubuntu, Alpine и слоёв Docker-образов. Также проводятся эксперименты, чтобы оценить целесообразность хранения набора хешей для дальнейшей сверки с ранее проанализированными файлами и понимания того, в каких пакетах и образах встречалось это содержимое.

В итоге разрабатываемое решение НТЦ «Фобос-НТ» позволяет находить расхождения ППК с формуляром и неправдоподобные записи в ППК.

## ИИ И БЕЗОПАСНОСТЬ

Параллельно мы решаем задачу защиты самих ИИ-моделей, которые становятся объектами сертификации. НТЦ «Фобос-НТ» уже второе десятилетие защищает государственные информационные системы и объекты критической информационной инфраструктуры, и новым вызовом является необходимость защиты применяемых там моделей искусственного интеллекта. Если система обрабатывает персональные данные, иную информацию ограниченного доступа или реализует значимые функции оператора, то, согласно требованиям приказа ФСТЭК России от 11 апреля 2025 г. № 117, необходимо провести сертификацию программного обеспечения, обеспечивающего реализацию системы искусственного интеллекта.

**Наш рекордный ППК был размером 148 МБ и содержал больше трёх миллионов строк. Файл с ошибками тоже впечатлял объёмом и количеством строк. Поэтому поверх sbom-checker используется утилита sbom-report, которая сворачивает тысячи однотипных находок в классы, и уже её вывод мы передаём на анализ ИИ**

Добавление модели ИИ в СЗИ расширяет поверхность атаки: появляются векторы, которых у классических средств защиты просто не существовало. Вот некоторые из них:

- несанкционированный доступ или воздействие через модель при инференсе реализуются атаками уклонения, когда входные данные подбираются так, чтобы обмануть модель и заставить её выдать неверный результат;
- воздействие через обучающие данные или процесс обучения реализуется атаками отравления, при которых злоумышленник искажает обучающую выборку, влияя на поведение модели и внедряя скрытые закладки;
- утечка информации ограниченного доступа через ответы модели реализуется атаками на приватность, когда по ответам модели злоумышленник может восстановить сведения о данных обучения, используя модель как источник утечки.

Определить устойчивость к таким типам атак вручную весьма трудно, поэтому НТЦ «Фобос-НТ» в рамках индустриального партнёрства в сфере ИИ с ИСП РАН работает над двумя инструментами:

- MLM — платформа для управления жизненным циклом моделей. Каждая задача выполняется в отдельном контейнере, а все артефакты сохраняются, что позволяет в любой момент взять тот же набор данных, ту же версию модели и сценарий атаки, чтобы повторить испытание и получить идентичный результат.
- TAI — инструмент оценки устойчивости моделей. Он проверяет модель с помощью атак уклонения и отравления, а также атак на приватность, выдавая числовую оценку, которая показывает, как меняется качество модели при увеличении бюджета атаки, какая доля отравленных данных необходима для достижения цели или насколько успешно извлекается информация об обучающих данных.

Для защиты искусственного интеллекта в информационных системах мы учитываем разные сценарии его применения в составе программного обеспечения и оцениваем возможности инструментов по анализу защищённости таких решений. Поэтому результаты нашей апробации используются коллегами из ИСП РАН для улучшения и дальнейшего развития инструментов.

## ПОДВОДЯ ИТОГ

Подводя итог, отметим: ИИ в испытательной лаборатории — не вместо эксперта, а ради него.

Технология берёт на себя рутинные и трудоёмкие операции, высвобождая ресурсы для решения более сложных задач. А на базе накопленного опыта НТЦ «Фобос-НТ» формирует технологичное решение, полезное и для экспертов лаборатории, и для разработчиков. Приглашаем к неформальному обсуждению деталей и обмену практиками 9–11 декабря на открытой конференции ИСП РАН 2026.

# СЕРТИФИКАЦИЯ СЗИ НАЧИНАЕТСЯ ДО ИСПЫТАНИЙ

**Андрей КУЗНЕЦОВ**
директор департамента исследований по выявлению уязвимостей и недекларируемых возможностей НТЦ «Фобос-НТ»

**Мы уже не раз писали, что у разработки безопасного ПО (РБПО) нет альтернативы, когда речь идёт о сертификации средств защиты информации (СЗИ). Но и само РБПО не стоит на месте. Появляются новые вызовы и инструменты, у разработчиков другой ритм релизов, и испытательной лаборатории приходится перестраиваться вслед за ними.**

Самые дорогие ошибки сертификации закладываются задолго до испытаний. «Фича», добавленная за месяц до подачи, расширяет поверхность атаки. Документация, написанная задним числом, расходится с кодом. Вскрывается это уже в сертификационном цикле, когда каждая переделка увеличивает сроки и расходы. Поэтому одним из главных принципов нашей лаборатории является формирование в команде продуктовой разработки чёткого понимания всех требований нормативных и методических документов. Лаборатория подключается, пока продукт ещё создаётся и зачастую «ревьювит фичреквесты», например, оценивая их влияние на поверхность атаки. На испытания заявитель приходит с продуктом, который к ним готов — как этого и требует ФСТЭК России «по классике». После начала испытаний разработчик спокойно, планово помогает в их организации, а не латает весь накопленный техдолг в последний месяц. Это экономит нервы и деньги всем сторонам.

## КЛАССИЧЕСКИЕ ИНСТРУМЕНТЫ

Используемые при сертификации инструменты анализа ПО не должны пылиться на полке после испытаний, а должны быть встроены в жизненный цикл разработки — именно этого требует Методика выявления уязвимостей (ВУ) и недекларированных возможностей (НДВ) [1]. Испытания, проводимые в такой парадигме, проверяют, что процесс работает и даёт результат, а не что перед подачей кто-то разово запустил анализатор. Мы как лаборатория не верим в «инструменты для сертификации», но считаем правильным использовать инструменты технологических лидеров. Для себя мы выбрали продукты ИСП РАН и CodeScoring — наших коллег и технологических лидеров отрасли, результатам и циклу развития которых доверяем, а ставку делаем на комплексные конвейеры РБПО, которые строим вместе с ведущими core-разработчиками инфраструктуры (ИСП РАН, «Базис», «Базальт СПО», «Флант»).

## ИНСТРУМЕНТЫ НОВОГО ВРЕМЕНИ

Классический набор инструментов позволяет составить представление о качестве кода. Но, согласно новым требованиям [1], нельзя обходить стороной и анализ поведения ПО в средствах оркестрации с отслеживанием реальных потоков данных между компонентами, а также исполняемого в браузере кода, который у многих СЗИ давно стал полноценной частью поверхности атаки. Для анализа прав доступа и динамического анализа архитектуры мы используем Лантри, для браузерного кода DPA FAST Analyzer. С ними испытания идут «на результат», и это уже рабочая практика, а не эксперимент.

## ФАЗЗИНГ XXI ВЕКА

Ещё одним из направлений нашей деятельности является построение фаззинг-конвейеров в соответствии с методологией Центра исследований безопасности системных компонент ФСТЭК России и ИСП РАН [2]. А пока команды привыкают к тому, что фаззинг плотно вошёл в жизненный цикл разработки — и чтобы конвейер не превратился в свалку крешей — мы готовы взять на себя часть работ по тестированию и триажу ПО (но не больше 30% — за основной объём всегда должен отвечать разработчик). А для самых передовых клиентов, которым интересно нахождение хардовых багов, в совокупности с подготовкой материалов испытаний, мы предлагаем работу по схеме баг-баунти поверх фаззинга. Идея такая: базовая фазз-цель стоит дешевле, но если помимо бага мы приносим работающий PoC, заказчик доплачивает по критичности уязвимости. Заказчик платит за подтверждённую опасность, а не за объём сырых падений. Именно через эту схему мы заходим на сложные задачи, в частности на Application Pentest.

## И ИИ

Нам самим интересно, как будут испытывать СЗИ с ИИ, поэтому мы участвуем в рабочих группах ТК 362 и других РГ. Мы индустриальные партнёры ИСП РАН и вкладываемся в новые инструменты РБПО для ИИ, то есть в MLSecOps [3].

Требования к ИИ в СЗИ уже на подходе, поэтому параллельно мы начинаем обкатывать подходы к определению ПА, к выявлению угроз и уязвимостей, специфичных для ИИ, и к видам анализа, которые их обнаруживают. Пробуем на тех СЗИ, где ИИ уже есть, в добровольном порядке и до выхода требований ФСТЭК России. Когда требования выйдут, мы хотим подойти к ним с боевым опытом, а не с чистого листа.

## ИСТОЧНИКИ

[1] https://cs.groteck.com/IB_4_2026/66/
[2] https://portal.linuxtesting.ru/
[3] Статья «Не вместо эксперта, а ради него» на стр. 92