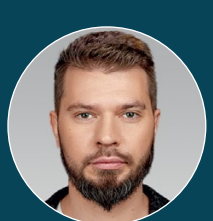
BIS JOURNAL
ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ БИЗНЕСА
№1 (56) 2025

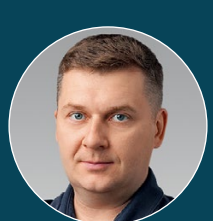
Владимир МАТЮХИН
По случаю юбилея
Пионеру российской ИБ – 80 лет!
→ 106

Сергей ГУСЕВ
АО «Северсталь Менеджмент»
«Броня крепка, а будет крепче!»
→ 4

БЕЗОПАСНАЯ РАЗРАБОТКА ЭВОЛЮЦИЯ ПО ГОСТУ



Алексей СМИРНОВ
CodeScoring
Open source: безопасное использование
→ 40



Антон БАШАРИН
AppSec Solutions
Безопасность – это ваша прибыль!
→ 44

Дарья ФИГУРКИНА
Swordfish Security
Экспертов с низкой образовательной базой заменит ИИ
→ 48

Сергей ПАЗИЗИН
научный редактор
BIS Journal

Долгое время вопросы обеспечения информационной безопасности промышленных предприятий в части сегмента операционных технологий практически не попадали в фокус внимания профильных конференций и периодики. Безусловным лидером и двигателем развития информационной безопасности являлся финансовый сектор, который всегда был привлекателен для киберпреступников, так как позволял обеспечить прямую монетизацию их усилий. Также определенное внимание уделялось сегменту управления предприятием, в котором обрабатываются персональные данные, платежи, информация, составляющая коммерческую тайну.

Последние несколько лет ситуация кардинально изменилась. В операционном сегменте промышленных предприятий расширяется процесс автоматизации, возрастает количество связей с сегментом информационных технологий, широко внедряется роботизация и устройства интернета вещей. Таким образом, операционный сегмент становится более понятен и доступен для злоумышленников. Кроме того, в связи с последними политическими событиями, появился постоянный заказчик для деструктивной деятельности киберпреступников.

С учетом сказанного, в этом году журнал принимает активное участие в организации и проведении конференции «Цифровая устойчивость промышленных систем» 27–28 февраля 2025 г., а тему конференции мы взяли в качестве заглавной темы номера. Наша задача — познакомить читателей с актуальными задачами и особенностями обеспечения информационной безопасности на промышленных предприятиях.

Одной из актуальных проблем обеспечения безопасности, в том числе промышленных предприятий, является задача импортозамещения не только средств защиты информации и системного и прикладного ПО, но автоматизированных систем управления технологическими процессами. Причем, импортозамещенное ПО должно удовлетворять самым высоким требованиям безопасности. Это привело к огромному интересу к методам и средствам разработки безопасного ПО, в том числе, изменению нормативного регулирования в этой области. Поэтому большое внимание в нашем номере мы уделили также теме разработки безопасного ПО.

BIS JOURNAL
ИЗДАЁТСЯ С 2010 ГОДА

УЧРЕДИТЕЛЬ-ИЗДАТЕЛЬ
ООО «Медиа Группа „Авангард“»
Адрес: 127473, Россия, Москва, 3-й Самотечный переулок, д. 21
Тел.: +7 495 921 42 44
Интернет-версия: ib-bank.ru/bisjournal
E-mail: bis@ib-bank.ru

Зарегистрирован 13 августа 2010 года Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзором) как «BIS Journal — Информационная безопасность банков».

Зарегистрирован 11 июля 2023 года Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзором) как «BIS Journal — Информационная безопасность бизнеса». Свидетельство ПИ № ФС77–85487 от 11.07.2023 г.

Подписан в печать 05.02.2025. Тираж 7000 экз.
Цена договорная. Отпечатан в типографии «Московский Печатный Двор», 123298, г. Москва, 3-я Хорошевская 18, корп. 1, офис 201А

 16+

«РБПО — не для ФСТЭКа. РБПО — для безопасности и качества вашего ПО».
Представитель ФСТЭК России

РБПО-2024. ИТОГИ И ПЕРСПЕКТИВЫ

АКТУАЛЬНОСТЬ ТЕМАТИКИ РБПО

Дмитрий ПОНОМАРЁВ
заместитель генерального директора — директор департамента внедрения и развития практик РБПО ООО НТЦ «Фобос-НТ», сотрудник ИСП РАН, преподаватель МГТУ им. Н. Э. Баумана

Если задаться целью проанализировать, какие именно слова были наиболее популярными в 2024 году в области, связанной с разработкой и поддержкой программ и систем, акроним РБПО с высокой вероятностью займёт лидирующую позицию[1]. Бесспорно, никуда не делся, хотя и несколько снизил темпы маркетинговой экспансии «искусственный интеллект». Резкий всплеск внимания к термину NGFW — все стремятся «застолбить поляну», правила работы на которой ещё толком не определены. Всё больше внимания обращается на микросервисные программные решения: «облака», сервисы банков и маркетплейсов и даже средства защиты информации — в контейнерном исполнении, в том числе функционирующие в кластерах под управлением kubernetes.

Но все эти понятия и слова в итоге подразумевают под собой определённые типы и особенности эксплуатации ПО. А где ПО — там и необходимость его создания и поддержки. Качественно, безопасно, управляемо и, что немаловажно, в соответствии с парадигмой действующей и перспективной регуляторики. «Комплаенс» — мощнейший стимул становиться лучше, особенно в современных условиях стратегической турбулентности и уже никому не требующих доказательства фактов непрестанных атак на отечественную IT-инфраструктуру. При этом необходимо отметить: несмотря на все сложности последних лет, мы не находимся в уникальной ситуации. Рост рынка киберпреступности в мире оценивается в «триллионы рублей в год»[2], степень информатизации становится всё выше, а атаки — всё изощреннее.

Реакцией на давление киберпреступности — как «частной», так и инициируемой на уровне недружественных государств (не все APT-группировки только лишь «про деньги») — является рост системности и проработанности различных требований в области информационной безопасности и, как следствие, рост технологий, инструментов и методик РБПО, а также вовлечённого в эти процессы сообщества инженеров, экспертов, чиновников и руководителей. Все эти факторы приводят к трансформации отечественных рынков ИБ и ИТ в целом. Парадигма «Внедрённые процессы РБПО — показатель зрелости компании и бизнес-преимущество XXI века» становится всё более распространённой, а в ряде сегментов — безальтернативной. Развиваются имеющиеся и возникают новые классы инструментов анализа. Пресса и профильные конференции пестрят анонсами о важности РБПО и предложениями продуктов и услуг. Безотносительно к тому, написаны они заинтересованными специалистами-инженерами или «ванильными маркетологами» с целью набить цену собственной компании, информационный поток становится всё более значимым, и из «темы для избранных» вопросы РПБО за единицы лет стали настоящим мейнстримом.

Эта статья пополнит копилку материалов «по теме» и, надеемся, окажется полезной читателям. В ней я постарался осветить некоторые типовые вопросы, возникающие у разработчиков — лицензиатов ФСТЭК России — при внедрении инструментов и развитии практик РБПО и последующем выводе собственных решений на сертификацию. Материал можно рассматривать в качестве логического продолжения опубликованной в 2024 г. статьи «6 мифов о безопасной разработке и сертификации ПО»[3].

[1] Во время написания статьи вышел материал (https://habr.com/ru/news/860268/), не подтверждающий данную гипотезу для сферы ИТ в целом. Тем не менее автор оставляет за собой право считать утверждение верным в отношении компаний-лицензиатов отечественных регуляторов.

[2] https://www.statista.com/hart/28878/expected-cost-of-cybercrime-until-2027/

[3] https://cs.groteck.com/IB_2_2024/60/

ТИПОВЫЕ «ПРОБЛЕМЫ» ЗАЯВИТЕЛЯ

Примерно с 2020 года развитие отрасли создания средств защиты информации можно охарактеризовать в том числе следующими тезисами:

- растёт число лицензиатов ФСТЭК, равно как и число заявителей на сертификацию СЗИ;
- растёт число малых команд, приходящих на сертификацию с одним небольшим продуктом и имеющих околонулевой опыт в сертификации; при этом многие команды ничего не слышали о требованиях РБПО и считают, что сертификация — это «что-то записанное в контракт по остаточному принципу, сделаем за 3–6 месяцев, не напрягаясь»;
- всё больший процент кодовой базы СЗИ составляют компоненты с открытым исходным кодом; у некоторых решений он достигает 100%, а на долю разработчика остаются компоновка, конфигурирование, маркетинг и поддержка;
- всё большую долю кодовой базы составляют интерпретируемые языки программирования / языки с «управляемой памятью» (Python, JS, C#, Java и т.д.);
- более половины приходящих на сертификацию продуктов составляют «СЗИ в контейнерном исполнении[4]», выполняющиеся в контексте средств контейнеризации (в том числе оркестратора kubernetes).

Наряду с иными веяниями времени вышеуказанное приводит к типовым «проблемам», возникающим у разработчиков, впервые оказывающихся в гостях у въедливой испытательной лаборатории:

- разработчик не умеет собирать своё решение из исходников, использует бинарные пресобранные компоненты из произвольных источников; нарушение базового принципа «весь код СЗИ === ваш код»;
- не ведётся контроль наличия уязвимых версий компонентов либо ведётся по остаточному принципу, поскольку: «их так много, мы же всё не поправим», «да оно не эксплуатируемо», «да наше ПО будет применяться только в доверенном контуре, никаких внутренних врагов нет»;
- у разработчика СЗИ в контейнерном исполнении, а значит: «у нас всё в контейнерах, значит, всё безопасно и вообще, упадёт — переподнимем», «что это вы такое придумали — контейнеры самому собирать и анализировать код компонентов... Это же Alpine с DockerHub, там всё отлично и безопасно»;
- разработчики СЗИ относятся к пакетной базе репозиториев отечественных сертифицированных ОС (~50 000 пакетов) так же, как к самому сертифицированному дистрибутиву (~300 пакетов): «разработчик ОС сказал, что в репе тоже всё доверенное, берите, чего в дистрибутиве не хватает».

Ещё одной типовой проблемой организации и контроля процессов РБПО является «разнобой» в используемых инструментах анализа, в том числе между разработчиками СЗИ и испытательными лабораториями. Часть разработчиков клюёт на активность маркетологов и приобретает за большие деньги «самый раскрученный», но не самый качественный инструмент, часть — принципиально экономит на платных инструментах, используя открытые, безотносительно к качеству их работы, зачастую только ради формального выполнения требования о проведении соответствующего вида анализа. Наиболее жаркие «дискуссии» в отрасли традиционно идут вокруг средств поиска известных уязвимостей и ошибок конфигурирования компонентов системы. Второе же место, бесспорно, принадлежит проблематике выбора инструментов статического анализа исходного кода. Данная практика анализа является широко распространённой и одной из базовых при построении процессов РБПО. Неслучайно требования ФСТЭК России к процессу статического анализа в части методики проведения сертификационных испытаний, равно как и в части комплектования испытательных лабораторий инструментами анализа, являются одними из наиболее проработанных.

Вышеуказанные факторы в совокупности с существенной разницей в компетенциях испытательных лабораторий, а также отсутствием прямых и конкретных разъяснений в нормативно-правовой базе по каждому возможному подвопросу (парадигма «писать рамочные требования — рассчитывать на сознательность и заинтересованность разработчика в безопасности и качестве», к сожалению, работает далеко не всегда) приводят к частому непониманию разработчиками СЗИ, «что именно, как и на какую глубину» требуется выполнить. Что, в свою очередь, порождает риски неопределённости для бизнеса, зависящие во многом от позиции конкретных экспертов, их персональных трактовок требований регулятора.

АКТУАЛИЗАЦИЯ РЕГУЛЯТОРИКИ И СООБЩЕСТВО РБПО

Ответом на вызовы времени явилось активное развитие в 2024 году нормативно-правовой базы ФСТЭК России. Ниже кратко описаны наиболее значимые изменения и сопровождавшие их события.

1. Информационным письмом «О порядке испытаний и поддержки безопасности средств защиты информации» изменён порядок исследования критичных компонентов с открытым исходным кодом при подготовке и проведении испытаний. При подаче заявки на сертификационные испытания заявители должны предоставить Перечень Программных Компонентов (ППК aka SBoM) в составе СЗИ и далее осуществлять их поддержку безопасности. Глубина и качество анализа компонентов, равно как и распределение нагрузки по анализу компонентов между разработчиками, определяются в рамках консорциума Центра исследований безопасности системного ПО ФСТЭК России и ИСП РАН[5].

2. Информационным письмом (от 25.10.2024)[6] уточнён порядок анализа компонентов, реализую-

[4] https://clck.ru/3GAsaP

[5] https://portal.linuxtesting.ru/index.html#regulations

[6] https://clck.ru/3GAsim

щих функции серверов приложений, веб-серверов и интерпретаторов. Функции безопасности данных компонентов должны быть корректно заявлены в формулярах СЗИ и проверены в ходе испытаний, при этом разработчик СЗИ должен сформулировать рекомендации по безопасной настройке указанных компонентов, а также минимизировать поверхность атаки на компонент посредством удаления из комплекта поставки функциональных возможностей, не предполагаемых к использованию. В первую очередь данные требования затрагивают разработчиков ОС, традиционно включающих в составы своих дистрибутивов значительное число таких компонентов[7].

3. На ноябрьских сборах ФСТЭК России с разработчиками СЗИ, испытательными лабораториями и органами по сертификации значимый акцент сделан на том, что компоненты СЗИ, упакованные в контейнеры различных форматов, в полной мере подлежат соответствующим уровню доверия проверкам в случае вхождения в состав поверхности атаки / реализации ими функций безопасности. Контейнеры должны формироваться из доверенной компонентной базы либо в полном объёме компоноваться разработчиком самостоятельно, с полной ответственностью за безопасность исходного и исполняемого кода.

4. В ноябре ФСТЭК России запустила программу аттестации экспертов испытательных лабораторий. Типовые задания аттестации предполагают не только знание нормативно-правовой базы регулятора, но и — в первую очередь — проверку навыков эксперта в выполнении различных видов статического и динамического анализа. Основной упор в процессе аттестации делается именно на выполнение практических заданий, что лишний раз подчёркивает факт отхода регулятором от примата вопросов «бумажной безопасности» к верховенству вопросов практического РБПО.

5. Принятый в 2024 году ГОСТ Р 71207–2024 (статический анализ программного обеспечения)[8] фактически явился первым из целого семейства ГОСТ в области РБПО, запланированных к принятию в 2024–2026 годах. Одной из задач данного ГОСТа является регламентация процесса оценки качества инструмента статического анализа. Анонс испытаний отечественных статических анализаторов был сделан заместителем начальника 2 управления ФСТЭК России И.С.Гефнер[9] в ходе Открытой конференции ИСП РАН 11 декабря[10]. В настоящий момент осуществляется формирование жюри испытаний, комплекта тестов. К участию в испытаниях приглашаются все отечественные компании — разработчики статических анализаторов, а также заинтересованные участники сообщества. В качестве инфраструктурной основы организации тестирования предполагается использование наработок, полученных в рамках создания Унифицированной среды безопасной разработки ПО[11].

6. Сообщество РБПО центра компетенций ФСТЭК России и ИСП РАН[12] активно развивается и прирастает новыми участниками. Целями сообщества являются:

- популяризация системного, фундаментально-инженерного подхода к организации безопасной и качественной разработки и подготовке кадров;
- популяризация отечественных технологий и школ, в том числе в части лучших практик и регуляторики; поддержка и продвижение отечественной «культуры»;
- формирование «грибницы» доверенных, кросс-верифицированных горизонтальных связей. Ядро сообщества составляют «технари» (бизнесмены, инженеры, чиновники). В сообществе не приветствуется «*маркетологическая маркетология*».

Крупнейшая ежегодная встреча участников сообщества состоялась на полях уже упомянутой выше Открытой конференции ФСТЭК России и ИСП РАН.

Основные информационные ресурсы сообщества в «Телеграме», посвящённые вопросам инструментального анализа, объединения усилий по анализу разделяемой компонентной базы, организации просветительских мероприятий, превысили число в 1500 участников. Узнать подробности про семейство чатов сообщества вы можете, перейдя по приведённому QR-коду или по ссылке[13]:

ПОДВОДЯ ИТОГ

2024-й выдался чрезвычайно насыщенным на различные события в области ИТ, ИБ и регуляторики. «За скобками» статьи остались такие важнейшие для отрасли вопросы, как, например, сертификация процессов РБПО на соответствие обновлённому ГОСТ Р 56939–2024 или публикация проекта обновлений 17-го приказа и многие другие. Реальное влияние анонсированных и принятых решений в полной мере отрасль ощутит в 2025–2026 годах. Со своей стороны традиционно могу пожелать читателям всех рангов как можно скорее осознать безальтернативность парадигмы безопасной и качественной разработки ПО в современных реалиях, трансформации отечественной регуляторики в технологичный и полезный инженеру свод рекомендаций и требований, а также тезис о том, что «внедрённые процессы безопасной и качественной разработки — гарантия эффективного прохождения сертификационных испытаний и конкурентное преимущество XXI века»!

[7] https://clck.ru/3GAsaP

[8] https://protect.gost.ru/document1.aspx?control=31&baseC=6&page=3&month=2&year=2024&search=&id=257752

[9] https://t.me/sdl_community/7343

[10] https://www.isprasopen.ru/

[11] https://www.tbforum.ru/hubfs/Digital/SS/SS_ADAPT/TBF24_14-02-24_Падарян.pdf

[12] https://cs.groteck.com/IB_3_2023/40/

[13] https://t.me/sdl_community

ЗАКОНОДАТЕЛЬНЫЕ ИНИЦИАТИВЫ В ОБЛАСТИ РБПО

ТРЕНДЫ И ПРАКТИКА. ОБЗОР

Альбина АСКЕРОВА
руководитель направления взаимодействия с регуляторами Swordfish Security

ВВЕДЕНИЕ: ЧТО ЕСТЬ В ОБЛАСТИ РЕГУЛИРОВАНИЯ РБПО

Практика РБПО давно вышла за рамки традиционного DevOps с интегрированной безопасностью. Сегодня это уже полноценная идеология, в которой безопасность выступает как правило хорошего тона, часть этикета и гигиены.

В эволюционном развитии такого подхода всё происходит по классическим законам экономики: спрос рождает предложение. Если ранее большинство экспертов фокусировались на анализе кода, то сейчас мировоззрение меняется, и в поле зрения попадают и те аспкеты, которые требуют проведения ИБ-проверок — от проектирования архитектуры и бизнес-логики до харденинга инфраструктуры и анализа цепочки поставок ПО. Может ли система изначально быть идеальной, или жизненные циклы создания ПО таковы, что хороший результат можно достичь только при комплексном применении всех практик безопасной разработки?

С таким широким запросом на спектр практик мы обращаемся к регуляторике и изучаем, что уже существует в этой области. Порой эксперты не замечают требования, если напрямую не указано: «Чтобы построить РБПО — делай раз, два, три». А по сути, если знать, из чего состоит РБПО и в каких документах оно описано, то можно найти множество регуляторных требований и рекомендаций о том, как правильно выстраивать процессы.

Итак, посмотрим.

Наши основные регуляторы в данном вопросе — ФСТЭК России и Банк России. Есть ещё отраслевые требования, но они, как правило, издаются как ответвление к основным, например, требования Минцифры России к госсистемам, размещаемым на ЕЦП «ГосТех», или требования Минэнерго к объектам электроэнергетики.

Помимо регуляторных, есть требования, утверждаемые президентом, — федеральные законы, указы.

Также существуют документы общего рекомендательного характера — ГОСТы (они являются рекомендательными до момента, пока о необходимости их исполнения не будет сказано в обязательных к исполнению документах).

Кроме «источника» нормативного документа, регулирование можно также классифицировать по отраслям. Обычно специалисты по ИБ именно так и формулируют свой запрос, так как решают задачу по защите объектов в конкретной сфере экономики.

Можно выделить несколько ключевых направлений регулирования: для финтеха (кредитные и некредитные организации и ряд других организационно-правовых форм в финансовой сфере), для объектов КИИ (критической информационной инфраструктуры), для ГИС (государственных информационных систем), для информационных систем обработки ПДн (персональных данных).

Важно отметить, что одна сфера регулирования не исключает другую и при выборе набора требований необходимо знать всё об объекте — его бизнес-задачи, обрабатываемые данные и их объём, нюансы взаимодействия с внешними системами и многие другие аспекты. Может получиться так, что в отношении объекта нужно принимать меры защиты и для финтеха, и для КИИ, и для ПДн, и для ГИС.

Иногда компании планируют сертифицировать свои программные продукты во ФСТЭК России, и этот процесс предполагает проверку внедрения процедур безопасной разработки ПО.

Также важна информация о том, где будут размещаться вычислительные ресурсы. Если объект будет работать на платформе ГосТех, то для ГИС необходимо будет соблюдать не только требования ФСТЭК России, но и Минцифры РФ.

Таким образом, для специалистов по безопасности нет единого универсального набора регуляторных требований — это всегда набор, формирующийся из особенностей защищаемого объекта.

Из описанного выше можно предложить некоторый алгоритм выбора требований из всего множества:

1. Является ли защищаемый объект КИИ и/или ГИС, и/или ИСПДн, и/или финтех?
2. Если несколько «да», то важно изучить и учесть особенности регулирования для каждого. Например, если программный продукт является объектом КИИ, это не всегда говорит о том, что он значимый. Также,

если приложение обрабатывает ПДн, нет необходимости без разбора выполнять все требования в полном объёме. В каждой отрасли есть свои «уровни безопасности», назовём таким общим термином разные наборы требований.

3. Если ГИС, то планируется ли размещать её на платформе ГосТех?

4. Если значимый объект относится к КИИ — это электроэнергетика или, может, АСУ ТП?

5. Есть ли задача по сертификации продуктов во ФСТЭК России?

6. Если финтех, то кредитная или некредитная организация? Или, может, бюро кредитных историй?

ОСНОВНАЯ ЧАСТЬ: РЕТРОСПЕКТИВА — ЧТО БЫЛО И ЕСТЬ В НПА РБПО

Теперь рассмотрим конкретные упоминания РБПО в регуляторике.

Общие нормы, которые применимы или могут быть опционально применимы к любой отрасли:

- Указ Президента РФ от 18.06.2024 № 529 «Об утверждении приоритетных направлений научно-технологического развития и перечня важнейших наукоёмких технологий»
- Федеральный закон от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации
- ГОСТ Р 56939–2024 «Разработка безопасного программного обеспечения. Общие требования»
- ГОСТ Р 15408–3–2013 «Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий»
- ГОСТ Р 58412–2019 «Разработка безопасного программного обеспечения. Угрозы безопасности информации при разработке программного обеспечения»
- Приказ ФСТЭК России от 03.04.2018 № 55 «Об утверждении Положения о системе сертификации средств защиты информации»
- Приказ ФСТЭК России от 02.06.2020 № 76 «Требования по безопасности информации, устанавливающие уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий»

Нормы для субъектов критической информационной инфраструктуры (КИИ):

- Указ Президента РФ от 01.05.2022 № 250 «О дополнительных мерах по обеспечению информационной безопасности Российской Федерации»
- Федеральный закон от 26.07.2017 № 187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации»
- Приказ ФСТЭК России от 14.03.2014 № 31 «Об утверждении Требований к обеспечению защиты информации в автоматизированных системах управления производственными и технологическими процессами на критически важных объектах, потенциально опасных объектах, а также объектах, представляющих повышенную опасность для жизни и здоровья людей и для окружающей природной среды»
- Приказ ФСТЭК России от 25.12.2017 № 239 «Об утверждении Требований по обеспечению безопасности значимых объектов критической информационной инфраструктуры Российской Федерации»

Нормы для систем обработки персональных данных:

- Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных»
- Приказ ФСТЭК России от 18.02.2013 № 21 «Об утверждении состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных»

Нормы для финтеха:

- ГОСТ Р 57580.1–2017 «Безопасность финансовых (банковских) операций. Защита информации финансовых организаций. Базовый состав организационных и технических мер»
- Положение Банка России от 17.04.2019 № 683-П «Об установлении обязательных для кредитных организаций требований к обеспечению защиты информации при осуществлении банковской деятельности в целях противодействия осуществлению переводов денежных средств без согласия клиента»
- Положение Банка России от 08.04.2020 № 716-П «О требованиях к системе управления операционным риском в кредитной организации и банковской группе»
- Положение Банка России от 20.04.2021 № 757-П «Об установлении обязательных для некредитных финансовых организаций требований к обеспечению защиты информации при осуществлении деятельности в сфере финансовых рынков в целях противодействия осуществлению незаконных финансовых операций»
- Положение Банка России от 15.11.2021 № 779-П «Об установлении обязательных для некредитных финансовых организаций требований к операционной надёжности при осуществлении видов деятельности»
- Положение Банка России от 12.01.2022 № 787-П «Об обязательных для кредитных организаций требованиях к операционной надёжности при осуществлении банковской деятельности в целях обеспечения непрерывности оказания банковских услуг»
- Положение Банка России от 25.07.2022 № 802-П «О требованиях к защите информации в платёжной системе Банка России»
- Положение Банка России от 17.10.2022 № 808-П «О требованиях к обеспечению защиты информации при осуществлении деятельности в сфере оказания профессиональных услуг на финансовом рынке в целях противодействия осуществлению незаконных финансовых операций...»
- Положение Банка России от 03.08.2023 № 820-П «О платформе цифрового рубля»
- Положение Банка России от 17.08.2023 № 821-П «О требованиях к обеспечению защиты информации при осуществлении переводов денежных средств и о порядке осуществления Банком России контроля за соблюдением требований к обеспечению защиты информации при осуществлении переводов денежных средств»
- Положение Банка России от 07.12.2023 № 833-П «О требованиях к обеспечению защиты информации

для участников платформы цифрового рубля»

◆ Методический документ Банка России от 2021 года «Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций»

◆ Стандарт Банка России СТО БР ФАПИ.СЕК-1.6-2024. Безопасность финансовых (банковских) операций. Прикладные программные интерфейсы обеспечения безопасности финансовых сервисов на основе протокола OpenID Connect

◆ Стандарт Банка России СТО БР ФАПИ.ПАОК-1.0-2024. Безопасность финансовых (банковских) операций. Прикладные программные интерфейсы. Обеспечение безопасности финансовых сервисов при инициации OpenID Connect клиентом потока аутентификации по отдельному каналу.

◆ Методические рекомендации Банка России от 30.09.2024 № 16-МР по организации взаимодействия информационных систем организаций финансового рынка с инфраструктурой, обеспечивающей информационно-технологическое взаимодействие информационных систем, используемых для предоставления государственных и муниципальных услуг и исполнения государственных и муниципальных функций в электронной форме

◆ Методические рекомендации Банка России от 22.01.2025 № 2-МР по проведению тестирования на проникновение и анализа уязвимостей информационной безопасности объектов информационной инфраструктуры организаций финансового рынка

Предметные нормы для сферы энергетики:

◆ Приказ Минэнерго России от 26.12.2023 № 1215 «Об утверждении дополнительных требований по обеспечению безопасности значимых объектов критической информационной инфраструктуры, функционирующих в сфере электроэнергетики, при организации и осуществлении дистанционного управления технологическими режимами работы и эксплуатационным состоянием объектов электроэнергетики из диспетчерских центров субъекта оперативно-диспетчерского управления в электроэнергетике»

Требования для государственных информационных систем:

◆ Приказ ФСТЭК России от 11.02.2013 № 17 «Об утверждении требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах»

◆ Методические рекомендации по обеспечению безопасности при разработке программного обеспечения с использованием компонентов Единой цифровой платформы «ГосТех» (утверждено протоколом Президиума Правительственной комиссии от 08.12.2022 № 54)

◆ Концепция разработки безопасного ПО на платформе «ГосТех» (утверждено протоколом Президиума Правительственной комиссии от 04.05.2023 № 20)

ЧТО БУДЕТ МЕНЯТЬСЯ?

Указом от 18.06.2024 № 529 Президент России определил приоритетные направления научно-технологического развития и перечень важнейших наукоёмких технологий. «Безопасность получения, хранения, передачи и обработки информации» определена как приоритет научно-технологического развития, а в числе важнейших наукоёмких технологий мы видим «технологии создания доверенного и защищённого системного и прикладного программного обеспечения». Таким образом речь идёт о стратегическом направлении развития нашей страны, в котором РПБО занимает важное место.

Изменения происходят во всех сферах.

Произошедшее уже изменение — это пересмотр в полном объёме базо-

вого ГОСТ Р 56939–2024. Вместо 9 мер по разработке безопасного ПО, которые были с 2016 года, вводится 25 мер, которые в сумме содержат более 100 требований к процессу РБПО.

Среди нового перечня мер выделяются те, которые, как правило, требуют экспертных компетенций в области безопасной разработки: экспертиза и статистический анализ исходного кода, динамический анализ кода программы, использование безопасной системы сборки программного обеспечения, обеспечение безопасности используемых секретов, проверка кода на предмет внедрения вредоносного ПО через цепочки поставок и ряд других.

В развитие ГОСТ 56939-2024 сейчас ФСТЭК и экспертное сообщество ведут работу над разработкой ГОСТ по методике оценке уровня реализации процессов разработки безопасного ПО. Данным ГОСТ планируется зафиксировать подходы к оценке соответствия процессов ГОСТ 56939.

Еще в скором времени ФСТЭК планирует выпустить ГОСТ по композиционному анализу ПО, в котором будут описаны и процессы, и требования к инструментам, и требования по формированию перечня программным компонентов (перечень зависимостей, библиотек и других элементов, имеющих отношение к продукту).

В новой редакции ФСТЭК России планирует выпустить приказ от 11.02.2013 № 17 (проект изменений опубликован на сайте ФСТЭК России 08.08.2024) о защите информации в госсистемах. Положения о внедрении практик безопасной разработки добавлены в явном, акцентированном виде. В частности, не допускается использование ПО без проведения работ по экспертизе, тестированию и (или) исследованиям исходного кода, в том числе без проведения статического и динамического анализа кода программ, а также композиционного анализа программного обеспечения.

Также важно, что регулятор планирует утвердить эти требования не только для систем в статусе ГИС, но и в целом для всех объектов, которые эксплуатируют госорганизации.

Также и Банк России планирует пересмотреть положения № 821-П и № 757-П. Операторы по переводу денежных средств, банковские платёжные агенты (субагенты), некредитные финансовые организации получат право не проводить контроль каждой версии ПО, если они имеют «заключение проверяющей организации о том, что реализуемые процессы разработки программного обеспечения и приложений включают меры обеспечения безопасного жизненного цикла разработки программного обеспечения и приложений». То есть безопасная разработка позволит избежать контроля ПО (в том числе, если это объект КИИ, судя по действующим проектам новых редакций положений) — это достаточно революционный подход.

Такой же концептуально новый подход ФСТЭК России уже утвердила в 2023 году приказом № 240, в котором допускается вместо сертификации новых версий СрЗИ иметь сертифицированный процесс разработки этого СрЗИ.

Также, думаю, и с принятием закона «об оборотных штрафах» за утечки ПДн всё больше внимания будет уделяться безопасности ПО, обрабатывающего ПДн.

ТОП-3 КЛЮЧЕВЫХ ТРЕНДА

Какие тренды безопасной разработки в России можно выделить с учетом вектора развития регуляторики:

1. Импортозамещение и доверие. Open Source по-прежнему популярен, но в современных реалиях для критических сфер в российской экономике использовать его небезопасно (недавний прецедент с ядром Linux яркий тому пример) , и наблюдается тренд перехода на отечественные решения и инструменты для реализации и автоматизации практик РБПО.

2. Повышение компетенций ИТ-специалистов. Относится к разработчикам, ИБ-специалистам, сетевикам и всем, кто обеспечивает эксплуатацию объекта, для которого внедряется РБПО.

Этот тренд, как результат осознанного перехода к концепции Secure by Design, свидетельствует и о максимальном «смещении влево» в обеспечении безопасности программных продуктов компаний.

3. Применение ИИ и машинного обучения. Использование технологий доверенного искусственного интеллекта для выявления уязвимостей, анализа поведения приложений и предотвращения атак. Машинное обучение помогает в прогнозировании угроз и автоматизации процессов тестирования на безопасность.

Ключевое слово тут «доверенный» — применением ИИ сейчас никого не удивишь, но можем ли мы на 100% довериться ему в защите своих критических ресурсов? Подходы и принципы «доверия» к ИИ, а также регулирование этой технологии сейчас являются практически основной повесткой всех дискуссий профессионального сообщества.

ЗАКЛЮЧЕНИЕ: НОРМАТИВНО-ПРАВОВОЕ РЕГУЛИРОВАНИЕ РБПО — НЕОБХОДИМО ДЛЯ ОБЕСПЕЧЕНИЯ КИБЕРУСТОЙЧИВОСТИ

Очень часто от регуляторики требуют меньше бюрократии, бумажной безопасности и больше практической пользы, описанной лаконично, просто и с конкретным набором действий.

Но, по сути, наши регуляторы уже давно уделяют этому много внимания, выпуская пояснения, методические материалы, да и в целом отвечая на прямые вопросы офлайн или онлайн, а также привлекая профессионалов и экспертов из коммерческих организаций для совместной работы над регуляторикой.

Безопасность всегда была и остаётся процессом, только сейчас точка старта смещается влево у всё большего числа компаний, а не только у крупнейших корпораций.

Государство в лице регуляторов задаёт и корректирует вектор развития процессов и технологий разработки безопасного ПО, помогая сохранить баланс между скоростью выпуска программных продуктов в эксплуатацию и защищённостью чувствительной информации.

СОВРЕМЕННЫЕ ПРАКТИКИ РБПО ДЛЯ ФИНАНСОВОГО СЕКТОРА

Андрей ПРОХОРЕНКО
руководитель отдела аудита и консалтинга
по безопасной разработке компании Swordfish

Госдума ужесточила санкции за утечку персональных данных. Теперь штрафы достигают 15 миллионов рублей, а в случае повторной утечки компании будут вынуждены платить уже оборотные штрафы. Эта новость особенно важна для бизнеса, который напрямую несёт ответственность не только за персональные данные, но и за финансы клиентов. Разберёмся, как избежать проблем и убытков при помощи грамотного построения безопасной разработки.

Финансовый сектор — один из самых регулируемых в мире, особое внимание уделяется защите финансовых активов физических и юридических лиц. С ростом зависимости бизнеса от количества используемых технологий, длинных цепочек поставок и цифровизации соблюдение требований по обеспечению безопасности приложений становится ещё более критичным, чем когда-либо.

Несоблюдение этих требований может привести к значительным финансовым потерям и потере доверия клиентов. Так, согласно исследованию IBM «Cost of Data Breach Report» за 2024 год средняя стоимость утечки данных по финансовому сектору в мире постоянно растёт и за последний год выросла на 3%. При этом суммарная стоимость потерь бизнеса в результате успешных атак выросла почти на 11% по сравнению с 2023 годом. Потери бизнеса включают в себя потерю выручки из-за простоя в работе сервисов, стоимость оттока активных клиентов и принесённый ущерб репутации (рис. 1).

Отказоустойчивость сервисов не единственный показатель надежности финансовых систем , важна и защита информации (персональные данные, финансовые операции, коммерческая тайна и др.). Так, Россия уже несколько лет находится в лидерах мирового антирейтинга стран по количеству утечек персональных данных. Об этом, кстати, свидетельствует рост предложений по украденным базам на теневом рынке.

К счастью, мы видим, что компании всё больше внимания уделяют практикам безопасной разработки и начинают активнее применять меры, руководствуясь наилучшими практиками. К слову о наилучших практиках: 20.12.2024 введён в действие ГОСТ Р 56939–2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования» — последняя редакция одного из основополагающих документов по разработке безопасного ПО (далее – РБПО). Компаниям, которые только начинают свой путь построения безопасной разработки, рекомендуем прежде всего ориентироваться именно на этот документ.

С ЧЕГО НАЧИНАЕТСЯ РАЗВИТИЕ РБПО?

В первую очередь организация должна определить перечень предъявляемых ей регулятором требований. В отличие от других секторов, финансовый сектор строго контролируется регулирующими органами. Необходимо применять сертифицированное ПО или ПО, которое соответствует ОУД (оценочному уровню доверия).

Следующим шагом будет аудит и оценка уровня зрелости текущих процессов РБПО. Аудит и оценку можно провести собственными силами, например, руководствуясь международными (BSIMM, OWASP SAMM, Microsoft SDL, NIST SP 800–64) и отечественными стандартами (ГОСТ Р 56939–2024, ГОСТ Р 71207–2024). При детальном рассмотрении можно заметить, что большая часть международных стандартов основывается на одной и той же базе, и лишь немногие содержат уникальные рекомендации и практики в зависимости от направленности конкретного стандарта. У каждого стандарта есть свои преимущества и недостатки, так как же нам понять, с чего начать?

Компании из финансового сектора могут сочетать элементы разных стандартов для создания дорожной карты развития инициатив РБПО, соответствующей их уникальным потребностям, и гибко адаптировать лучшие мировые практики к российским реалиям. Не у всех есть необходимый набор компетенций, а зачастую и время, чтобы разбираться предметно в вопросе. Тогда можно обратиться к экспертам.

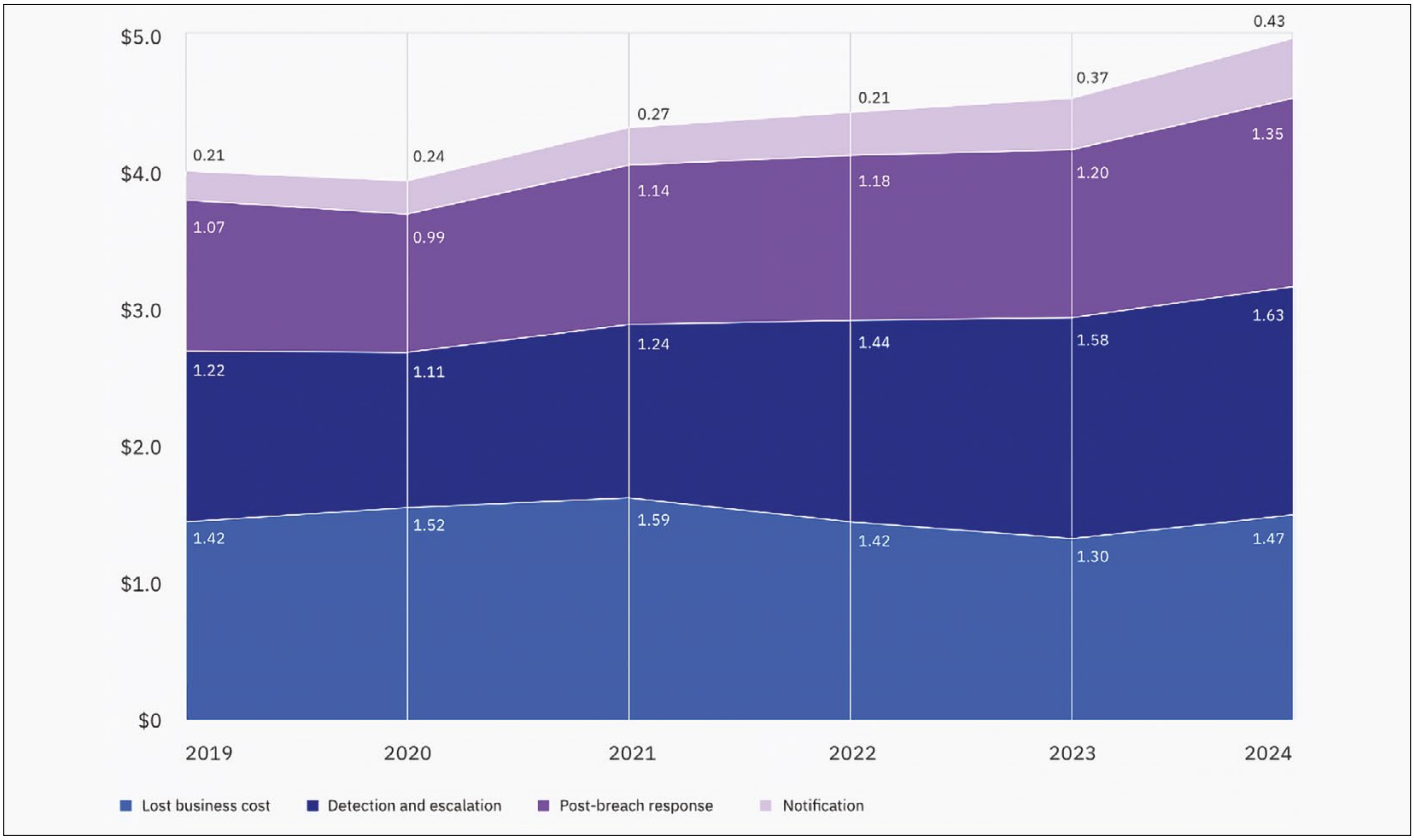


Рисунок 1. Средняя стоимость утечки данных (млн долл. США)

ПОДХОД К ВНЕДРЕНИЮ

По нашему опыту, внедрение вышеперечисленных стандартов или их комбинаций базируется на трёх основных элементах:

- технологии;
- люди;
- процессы.

Из этих элементов складывается корпоративная культура безопасной разработки, и в разрезе этих трёх ключевых элементов проводится аудит/оценка отправной точки.

Под технологиями мы подразумеваем прежде всего используемые в компании инструменты жизненного цикла безопасной разработки таких классов, как OSA, SCA, SAST, DAST, MAST, CS, ASPM.

При этом, инструмент бесполезен без знаний о том, как его правильно установить, настроить и интегрировать в производственный цикл. Именно люди обладают навыками, опытом и компетенциями, необходимыми для корректного и эффективного использования инструментов. Без опыта и экспертизы обработки результатов сканирования технический долг команд разработки будет только расти от релиза к релизу.

Одной из самых острых проблем сегодня является дефицит квалифицированных сотрудников, поэтому крайне важно не только оценивать текущие навыки и компетенции сотрудников, но и сформировать программу обучения специалистов. Обучение поможет облегчить введение в контекст безопасной разработки для новых сотрудников и закрыть отдельные пробелы в их знаниях.

Для обеспечения эффективной передачи экспертного опыта компаниям зачастую необходимо пересмотреть текущую ролевую модель сотрудников с целью определения недостающих ролей, а также формирования налаженного процесса взаимодействия между структурными единицами. Наличие знаний в головах, безусловно, хорошо, однако не хватает последней составляющей — чётко выстроенного процесса безопасной разработки, в основу которого ложится регламентирование и методологическое сопровождение. Без отработанных процедур невозможно добиться эффективной и результативной работы практик безопасной разработки.

ПОДЫТОЖИМ

Развитие практик РБПО актуально для финансовых организаций любой категории, но в разном объёме, есть индивидуальные особенности. При этом необходимо сфокусироваться на нескольких ключевых элементах:

- внедрении инструментов анализа безопасности;
- обучении сотрудников практикам безопасной разработки и вводе их в эксплуатацию;
- выстраивании и регламентации процессов, определяющих эффективное взаимодействие между командами.

На начальном уровне основные процессы и регламенты безопасной разработки в небольшой компании можно выстроить приблизительно за 1 год при должном уровне экспертного опыта специалистов. Достижение зрелого состояния безопасной разработки занимает, по нашему опыту, около трёх лет, но здесь всё может быть сильно индивидуально.

ПОИСК УЯЗВИМОСТЕЙ ПО ОТКРЫТЫМ ИСТОЧНИКАМ

В РАМКАХ ЦИКЛА РАЗРАБОТКИ БЕЗОПАСНОГО ПО

Виталий ВАРЕНИЦА
кандидат технических наук, доцент МГТУ имени Н. Э. Баумана

Сас АРУСТАМЯН
старший преподаватель МГТУ имени Н. Э. Баумана

Илья АНТИПОВ
аспирант МГТУ имени Н. Э. Баумана

Нелли МАГАКЕЛОВА
студентка Финансового университета при Правительстве РФ

В условиях стремительного роста технологий и всеобъемлющей зависимости от программного обеспечения с открытым исходным кодом безопасность становится приоритетом для разработчиков и организаций. В данной статье рассматриваются методы и инструменты анализа уязвимостей по открытым источникам, их применение для повышения уровня безопасности ПО и лучшие практики их интеграции в процесс разработки. Отдельное внимание отведено композиционному анализу, который представляет собой мощный инструмент для выявления уязвимостей в программном обеспечении, включая библиотеки и компоненты, используемые в разработке. Разработка безопасного программного обеспечения на данный момент в Российской Федерации применяется в соответствии с требованиями ГОСТ Р 56939–2024.

ВВЕДЕНИЕ

С увеличением сложности программных систем и распространением программного обеспечения с открытым исходным кодом (ПО с ОИС) уязвимости становятся всё более распространённой проблемой. Согласно исследованию, большинство атак на программное обеспечение происходит через уязвимости в сторонних библиотеках и компонентах. Это связано с тем, что доступность исходного кода позволяет злоумышленникам легко анализировать и выявлять слабые места, что, в свою очередь, ставит под угрозу безопасность многих систем, использующих такие решения. Открытость исходного кода, несмотря на её преимущества в обеспечении прозрачности и возможности проведения аудита, также создаёт риски для безопасности, если разработчики не придерживаются надлежащих практик защиты информации.

Анализ по открытым источникам информации позволяет разработчикам идентифицировать и управлять этими рисками. Различные источники указывают, что поиск уязвимостей в ПО с ОИС становится необходимым элементом цикла разработки безопасного программного обеспечения (SSDLC). Цель данной статьи — рассмотреть преимущества использования инструментов данного анализа в процессе разработки безопасного ПО.

МЕТОДИКА ПРОВЕДЕНИЯ АНАЛИЗА УЯЗВИМОСТЕЙ ПО ОТКРЫТЫМ ИСТОЧНИКАМ

Анализ сторонних компонентов программного обеспечения на предмет известных уязвимостей может быть проведён несколькими способами:

1. Ручной поиск уязвимостей по базам данных уязвимостей, который включает в себя поиск информации о компонентах в различных источниках информации, таких как:
   - ◆ национальные базы данных:
     - Банк данных угроз безопасности информации (БДУ ФСТЭК России);
   - ◆ зарубежные базы данных уязвимостей и недостатков (ошибок):
     - Common Vulnerabilities and Exposures (CVE);
     - CVE Details;
     - National Vulnerability Database;
     - Vulners.
   - ◆ публикации и тематические форумы;
   - ◆ результаты исследований, полученные сторонними исследователями;
   - ◆ официальный сайт производителей компонентов с открытым исходным кодом.

Этот подход позволяет идентифицировать известные уязвимости, однако требует значительных временных затрат, особенно при большом количестве компонентов и сложных зависимостях. Это делает его неэффективным для больших проектов или при необходимости быстрого анализа. Таким образом, эффективность и скорость ручного поиска уступают автоматизированным методам, и даже тщательный ручной поиск не гарантирует обнаружения всех уязвимостей.

2. Статический анализ исходного кода (SAST — Static Application Security Testing) — вид работ, который включает в себя поиск дефектов в исходном коде без его фактического выполнения.

3. Динамический анализ исходного кода (DAST — Dynamic Application Security Testing) — вид работ, который включает в себя поиск уязвимостей в процессе выполнения кода.

4. Нефункциональное тестирование — вид работ, который включает в себя проверки, не относящиеся к тестированию функциональных возможностей ПО. Данный термин был введён в ГОСТ Р 56939–2024 и в первую очередь является заменой понятия тестирования на проникновение. Однако, в отличие от классического пентеста, нефункциональное тестирование — более широкое понятие, включающее в себя также другие проверки, которые требуется проводить в соответствии с промышленными требованиями разработчика. В контексте анализа заимствованных компонентов авторы данной статьи подразумевают именно тестирование на проникновение.

5. Композиционный анализ (SCA — Software Composition Analysis) — вид работ, основанный на формировании перечня зависимостей ПО, определении лицензионной чистоты используемых компонентов и выявлении наличия уязвимостей и/или иных недостатков в зависимостях ПО.

Стоит дополнительно отметить, что статический анализ, динамический анализ и нефункциональное тестирование в отношении заимствованных компонентов требуют гораздо больших трудозатрат с точки зрения проведения, поскольку необходимо разработать уникальную методику тестирования каждого компонента. Соответственно, данные виды анализов целесообразно проводить на этапе тестирования программного обеспечения, когда состав заимствованных компонентов уже сформирован. Важно также подчеркнуть, что в процессе тестирования можно выявить zero-day-уязвимости. Это уязвимости, которые не были ранее известны разработчикам или производителям программного обеспечения, и, следовательно, на них отсутствуют патчи или средства защиты.

В свою очередь, ручной анализ по открытым базам данных и композиционный анализ рекомендуется интегрировать в жизненный цикл разработки программного обеспечения на более ранних этапах, включая стадии проектирования и разработки. Это поможет заведомо выявить проблемы, связанные с безопасностью, и не допустить использования в составе программных компонентов библиотек с подтверждёнными уязвимостями. Поскольку композиционный анализ является логическим продолжением анализа по открытым источникам, в дальнейшем стоит сделать акцент именно на нём.

Многие проекты используют пакетные менеджеры зависимостей (например, npm, maven, pip). Анализ списка зависимостей позволяет идентифицировать используемые компоненты и проверить их на наличие уязвимостей в вышеупомянутых базах данных. Инструменты для анализа состава компонентов помогают автоматизировать данный процесс.

Процесс разработки программного обеспечения, включающий использование компонентов с открытым исходным кодом, формирует определённую цепочку поставок, состоящую из аналогичных проектов с ОИС. Эти проекты могут ссылаться на дополнительные библиотеки с открытым исходным кодом, что приводит к образованию длинной последовательности зависимостей. В связи с этим возникает необходимость в проведении проверки этих зависимостей на наличие известных подтверждённых уязвимостей.

Этапы композиционного анализа традиционно включают в себя:

1. Идентификацию компонентов, которая определяет все внешние библиотеки и зависимости, используемые в проекте.

2. Выявление известных уязвимостей в этих компонентах с использованием различных баз данных уязвимостей, таких как CVE (Common Vulnerabilities and Exposures).

3. Проверка лицензионных соглашений используемых компонентов на предмет соответствия требованиям проекта.

Для прохождения первого этапа необходимо сгенерировать SBOM-файл (Software Bill of Materials) — файл, который содержит в себе спецификацию в виде машиночитаемого формата, в котором перечислены библиотеки и их версии.

Взаимодействие между программными компонентами определяется через граф зависимостей, вершины которого представляют компоненты, а рёбра — зависимости между ними. Зависимости делятся на прямые (direct dependencies) и транзитивные (transitive dependencies). Прямая зависимость устанавливается, если компонент A явно использует компонент B, что обычно отображается в метаданных проекта (например, package.json, requirements.txt). Транзитивная зависимость возникает, когда компонент A зависит от компонента B, а B, в свою очередь, зависит от компонента C. В этом случае C является транзитивной зависимостью для компонента A (рис. 1).

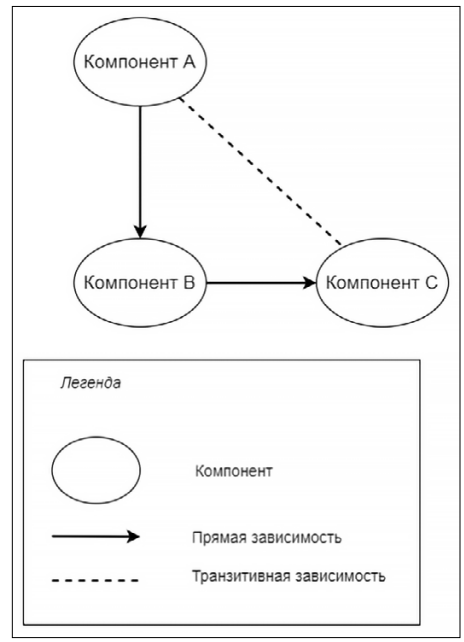


Рисунок 1. Транзитивная зависимость

ТРЕБОВАНИЯ РЕГУЛЯТОРОВ

Анализ компонентов открытого программного обеспечения на наличие известных уязвимостей является обязательным требованием множества отечественных и международных нормативных документов. Несмотря на повсеместную необходимость выявления уязвимостей в заимствованных компонентах (часто сводящуюся к поиску информации в открытых источниках для идентификации потенциальных угроз), явное указание на композиционный анализ как специфический метод проверки присутствует лишь в национальном стандарте ГОСТ Р 56939–2024 (таблица 1). Остальные рассмотренные стандарты (ГОСТ Р 56939–2016, ГОСТ Р ИСО/МЭК 15408-3-2013, «Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций», стандарт PCI DSS, NIST SP 800–171) описывают требования к анализу ПО с ОИС, однако не конкретизируют методологию.

Анализ требований, представленных в таблице далее по тексту, демонстрирует неоднородность подходов к анализу заимствованных компонентов по открытым источникам информации. В то время как необходимость такого анализа признаётся во всех рассмотренных документах, только ГОСТ Р 56939–2024 указывает на композиционный анализ как наиболее эффективный метод. Это свидетельствует на недостаток стандартизации в области оценки безопасности заимствованных компонентов, что может привести к неполному покрытию уязвимостей и повышенному риску компрометации системы. Дальнейшие исследования должны быть направлены на разработку более чётких и унифицированных методик, учитывающих специфику различных областей применения программного обеспечения.

Нормативный источник	Номер меры	Указание требований по композиционному анализу в явном виде
ГОСТ Р 56939–2016	п. 5.3.3.4, п. 5.3.3.5, п. 5.4.3.2, п. 5.4.3.3, п. 5.4.3.4, п. 5.6.3.4	Нет
ГОСТ Р 56939–2024	п. 5.12, п. 5.15, п. 5.16	Да
ГОСТ Р ИСО/МЭК 15408-3–2013	AVA_VAN.1.2E — AVA_VAN.5.2E, ACO_VUL.1.3E — ACO_VUL.3.3E	Нет
«Требования по безопасности информации, устанавливающие уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий» (утв. приказом ФСТЭК России от 2 июня 2020 г. № 76)	п. 12.2, п. 22.1	Нет
Методический документ «Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций» (утв. Банком России)	AVA_VAN.5.2E	Нет
Стандарт PCI DSS	Requirement 11.3.1.1	Нет
NIST SP 800–171	3.11.2	Нет

Таблица 1. Требования регуляторов

ИНСТРУМЕНТЫ, ПРИМЕНЯЕМЫЕ ПРИ КОМПОЗИЦИОННОМ АНАЛИЗЕ

Использование CI/CD позволяет автоматизировать процесс анализа зависимостей, что снижает вероятность возникновения человеческой ошибки и обеспечивает регулярное обновление информации о безопасности.

Для композиционного анализа наиболее популярными решениями являются OWASP Dependency Track и OWASP Dependency Check. Далее более подробно рассмотрим пример использования OWASP Dependency Track.

Для автоматизированного использования данного инструмента необходимо подготовить файл определённой структуры, в котором содержатся зависимые компоненты и информация о них (SBOM-файл). Необходимо выбирать утилиту для генерации файла в соответствии с используемыми языками программирования. Наиболее распространённая утилита генерации — CycloneDX (cdxgen).

После анализа SBOM-файла в OWASP Dependency Track можно отслеживать не только сами выявленные уязвимости (рис. 2), но и изменение количества уязвимостей в проекте, который инкрементально исследуется (рис. 3).

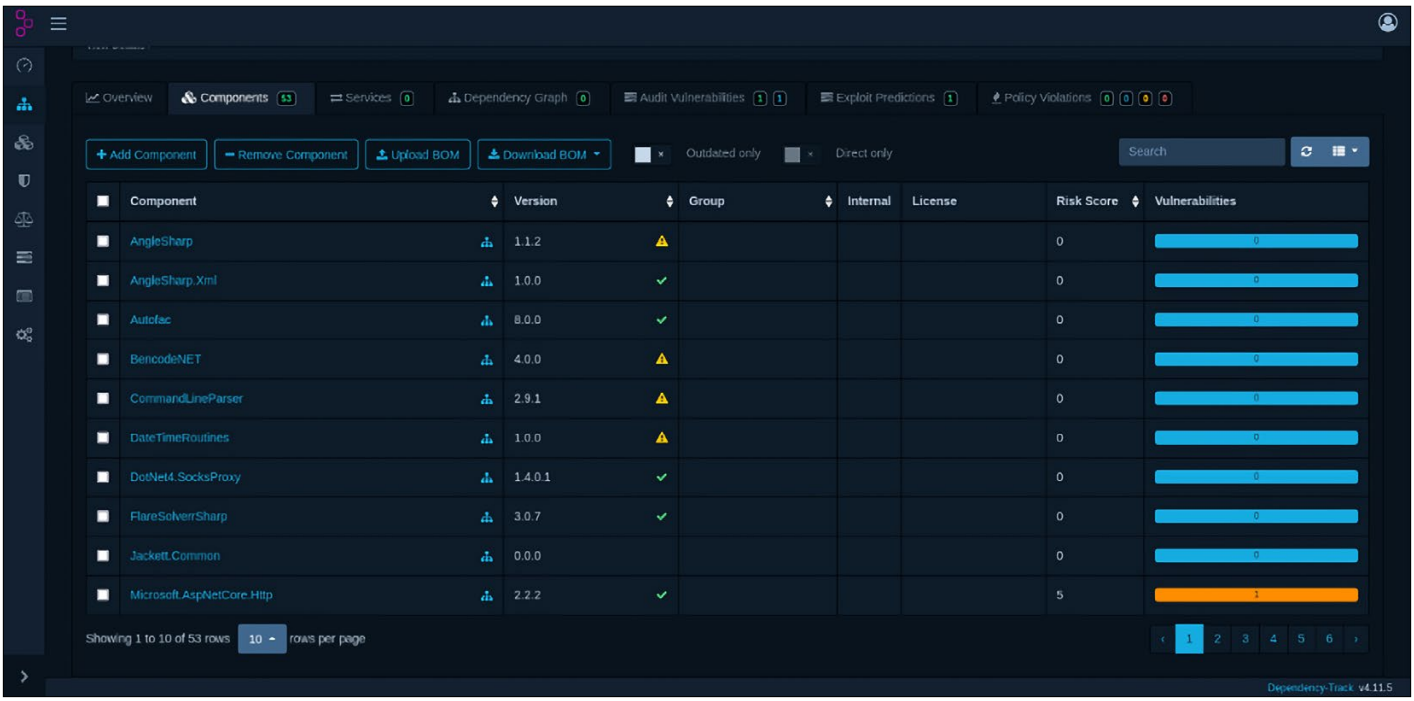


Рисунок 2. Выявленные уязвимости

Рисунок 3. График зависимости количества выявленных уязвимостей от версии проекта

ВЫВОД

Современная разработка с использованием ПО с ОИС требует особого внимания к вопросам безопасности. Применение методов и инструментов анализа уязвимостей, включая композиционный анализ, позволяет существенно снизить риски, связанные с использованием уязвимых библиотек и компонентов. Интеграция таких практик в процесс разработки способствует созданию надёжного ПО, соответствующего как международным стандартам, так и требованиям ГОСТ Р 56939–2024, актуального для Российской Федерации. Постоянное совершенствование подходов к обеспечению безопасности и внедрение инновационных решений позволяют разработчикам и организациям быть готовыми к вызовам стремительно меняющегося технологического ландшафта.

ПОСТРОЕНИЕ КОНТУРА РАЗРАБОТКИ БЕЗОПАСНОГО ПО

Виталий ВАРЕНИЦА
кандидат технических наук, доцент МГТУ имени Н. Э. Баумана

Сас АРУСТАМЯН
старший преподаватель МГТУ имени Н. Э. Баумана

Илья АНТИПОВ
аспирант МГТУ имени Н. Э. Баумана

Нелли МАГАКЕЛОВА
студентка Финансового университета при Правительстве РФ

В данной статье представлен подход к созданию контура разработки безопасного программного обеспечения, который применяется для создания инфраструктуры по разработке программного обеспечения, предусматривающей предотвращение появления проблем безопасности информации, а также их эффективное обнаружение и устранение. На текущий момент разработка безопасного программного обеспечения в Российской Федерации осуществляется в соответствии с требованиями национального стандарта ГОСТ Р 56939–2024.

ВВЕДЕНИЕ

В настоящее время разработка безопасного ПО становится всё более актуальной задачей. Это связано с тем, что всё больше разработчиков обращают внимание не только на оптимизацию и функциональные возможности продуктов, но и на их безопасность.

Эта проблема широко освещается в различных источниках. Например, в источниках подробно расписываются конкретные методики внедрения процессов разработки безопасного ПО. В работах специалистов отмечается необходимость минимизации различных проблем безопасности, которые возникают на этапе разработки.

ЖИЗНЕННЫЙ ЦИКЛ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (SDLC) И БЕЗОПАСНЫЙ ЖИЗНЕННЫЙ ЦИКЛ РАЗРАБОТКИ ПО (SSDLC)

SDLC (Software Development Life Cycle) — это процесс разработки программного обеспечения, который включает в себя последовательность этапов, начиная от первоначального предъявления требований и заканчивая поддержкой и обслуживанием готового продукта.

Цикл SDLC состоит из следующих этапов:

- анализ требований;
- планирование;
- проектирование и дизайн;
- разработка ПО;
- тестирование;
- развёртывание;
- эксплуатация.

Анализ требований подразумевает процесс выявления, определения и документирования требований к разрабатываемому продукту.

Планирование подразумевает процесс составления плана разработки программного обеспечения, который включает определение требуемых ресурсов для реализации плана разработки ПО, подготовки бюджета и формирование команд разработки.

Проектирование и дизайн подразумевает процесс продумывания всей архитектуры продукта в детализированном виде, а также анализ рисков.

Разработка программного обеспечения подразумевает процесс написания исходного кода, а также создания системы на основе требований и проекта архитектуры.

Тестирование подразумевает процесс проверки продукта на предмет несоответствий заявленным требованиям.

Развёртывание подразумевает процесс установки и настройки программного обеспечения, создание баз данных, конфигурирование сетевых настроек и других действий, необходимых для работы ПО.

Эксплуатация подразумевает процесс технической поддержки ПО, а также процесс обновления ПО в случае внесения изменений и (или) выявления ошибок в ходе эксплуатации.

Как можно заметить, в данном цикле основное внимание уделяется процессу разработки, который является ключевой задачей. Вопросы информационной безопасности не рассматриваются отдельно. Следовательно, для разработки безопасного программного обеспечения необходимо доработать этот цикл, интегрировав в него аспекты, связанные с безопасностью. Таким образом, цикл SSDLC (Secure Software Development Life Cycle) представляет собой модификацию цикла SDLC, в которую включены специфические методы, направленные на обеспечение безопасности. Более под-

Этап жизненного цикла ПО	Номер процесса	Описание процесса (ГОСТ Р 56939–2024)
Анализ требований	5.3	Формирование и предъявление требований безопасности к программному обеспечению
Планирование	5.1	Планирование процессов разработки безопасного программного обеспечения
	5.2	Обучение сотрудников
Проектирование и дизайн	5.4	Управление конфигурацией программного обеспечения
	5.6	Разработка, уточнение и анализ архитектуры программного обеспечения
	5.7	Моделирование угроз и разработка описания поверхности атаки
Разработка ПО	5.5	Управление недостатками и запросами на изменение программного обеспечения
	5.8	Формирование и поддержание в актуальном состоянии правил кодирования
	5.9	Экспертиза исходного кода
	5.10	Статический анализ исходного кода
	5.12	Использование безопасной системы сборки программного обеспечения
	5.13	Обеспечение безопасности сборочной среды программного обеспечения
	5.14	Управление доступом и контроль целостности кода при разработке программного обеспечения
	5.15	Обеспечение безопасности используемых секретов
	5.16	Использование инструментов композиционного анализа
	5.17	Проверка кода на предмет внедрения вредоносного программного обеспечения через цепочки поставок
Тестирование	5.11	Динамический анализ кода программы
	5.18	Функциональное тестирование
	5.19	Нефункциональное тестирование
Развёртывание	5.20	Обеспечение безопасности при выпуске готовой к эксплуатации версии программного обеспечения
	5.21	Безопасная поставка программного обеспечения пользователям
Эксплуатация	5.22	Обеспечение поддержки программного обеспечения при эксплуатации пользователями
	5.23	Обеспечение реагирования на информацию об уязвимостях
	5.24	Поиск уязвимостей в эксплуатирующемся программном обеспечении
	5.25	Обеспечение безопасности при выводе программного обеспечения из эксплуатации

Таблица 1. Дополнительные меры в цикле SSDLC

робно данные методы, а также сопоставление с текущими требованиями национального стандарта ГОСТ Р 56939–2024 указаны в таблице ниже (таблица 1).

Стоит принять во внимание, что некоторые из приведённых процессов невозможно отнести к конкретному этапу, поскольку они являются итеративными и могут пересекаться сразу с несколькими этапами жизненного цикла. Это связано с тем, что разработка программного обеспечения является цикличным процессом, где результаты одного этапа могут быть использованы для пересмотра предыдущих этапов.

Так, процесс «обучение сотрудников» по вопросам безопасности должен быть непрерывным процессом, охватывающим все стадии разработки ПО, который встраивается в корпоративную культуру. Это требует регулярных обновлений и пересмотров программ обучения, что выходит за рамки определённого этапа жизненного цикла. Поскольку первичное создание обучающих мероприятий осуществляется на ранних стадиях жизненного цикла ПО, данное требование было отнесено к этапу планирования.

Аналогично процесс «управление недостатками и запросами на изменение программного обеспечения» может возникать на разных этапах жизненного цикла ПО. Он включает в себя выявление, анализ и обработку недостатков, которые могут быть обнаружены как в процессе разработки (тестировщиками, разработчиками), так и в процессе эксплуатации (конечными пользователями).

ИНСТРУМЕНТЫ, ИСПОЛЬЗУЕМЫЕ В ПРОЦЕССЕ РАЗРАБОТКИ БЕЗОПАСНОГО ПО

Выполнение требований к разработке безопасного ПО подразумевает под собой комплекс организационных и технических мер. Помимо регламентов, которые необходимы для каждой из внедряемых мер, контур разработки подразумевает внедрение в инфраструктуру разработки инструментов, позволяющих эффек-

Этап жизненного цикла ПО	Номер процесса	Используемые инструменты	Пример инструментов
Анализ требований	5.3	Системы хранения документов проекта	Jira
Планирование	5.1	Системы планирования работ и управления задачами	Confluence
	5.2	–	–
Проектирование и дизайн	5.4	Аналогично п. 5.1, 5.3, 5.5, 5.9, 5.12, 5.21, 5.22	Аналогично п. 5.1, 5.3, 5.5, 5.9, 5.12, 5.21, 5.22
	5.6	Инструмент визуального моделирования и проектирования	Sparx Systems Enterprise Architect, Archimate
	5.7	Инструмент для определения поверхности атак	Natch
Разработка ПО	5.5	Системы управления изменениями	Jira, Bugzilla, Redmine, Azure DevOps, GitHub, GitLab, Bitbucket
	5.8	Линтеры	ESLint, Pylint
	5.9	Интегрированная среда разработки	Visual Studio Code, JetBrains IDE, Эшелониум
	5.10	Статические анализаторы	АК-ВС 3, Sonarqube, Svace, Horusec, Cppcheck, Clang SA
	5.12	Средства сборки	Apache Maven, Gradle, Make, CMake
		Инструмент контейнеризации	Docker
		Система для оркестрации	Kubernetes
	5.13	Аналогично п. 5.12	Аналогично п. 5.12
	5.14	Аналогично п. 5.5	Аналогично п. 5.5
	5.15	Статические анализаторы	АК-ВС 3, Sonarqube, Svace, Horusec, Cppcheck, Clang SA
		Инструменты сканирования для поиска секретов	Git-Secrets, gitLeaks, Git-all-secrets, Detect-secrets, Gittyleaks
	5.16	Инструменты композиционного анализа	OWASP Dependency-Check, OWASP Dependency-Track, CodeScoring
	5.17	Средства антивирусной защиты информации	Kaspersky Endpoint Security, Dr.Web Enterprise Security Suite, PT MultiScanner, Secret Net Studio 8
Тестирование	5.11	Инструменты динамического анализа	АК-ВС 3, Блесна
		Фаззеры	АК-ВС 3, AFL++, WinAFL, Crusher, Libfuzzer, Peach Fuzzer, Sharpfuzz, jazzer
	5.18	–	–
	5.19	Инструменты тестирования на проникновение	Burp Suite, OWASP ZAP, sqlmap, Metasploit
		Сетевые сканеры безопасности	Сканер-ВС, Nessus, Nmap
Развёртывание	5.20	Средства контрольного суммирования	ФИКС, ФИКС-Unix, Трафарет, ПИК Эшелон, gostsum
	5.21	Аналогично п. 5.20	Аналогично п. 5.20
		Инструменты резервного копирования	Citprotect, Rewind, BackupLABS, Veeam Backup Agent
Эксплуатация	5.22	Системы отслеживания и регистрации ошибок	Jira, Redmine, Azure DevOps, Bugzilla
	5.23	Аналогично п. 5.1, 5.5, 5.22	Аналогично п. 5.1, 5.5, 5.22
	5.24	Аналогично п. 5.8, 5.9, 5.11, 5.16, 5.19	Аналогично п. 5.8, 5.9, 5.11, 5.16, 5.19
	5.25	–	–

Таблица 2. Используемые инструменты в цикле SSDLC

тивно бороться с различными проблемами безопасности, возникающих по вине разработчика на этапе создания продукта. В таблице ниже (таблица 2) представлен перечень инструментов, требуемых для успешной реализации процессов РБПО.

СОСТАВ КОМАНД, НЕОБХОДИМЫХ НА КАЖДОМ ЭТАПЕ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

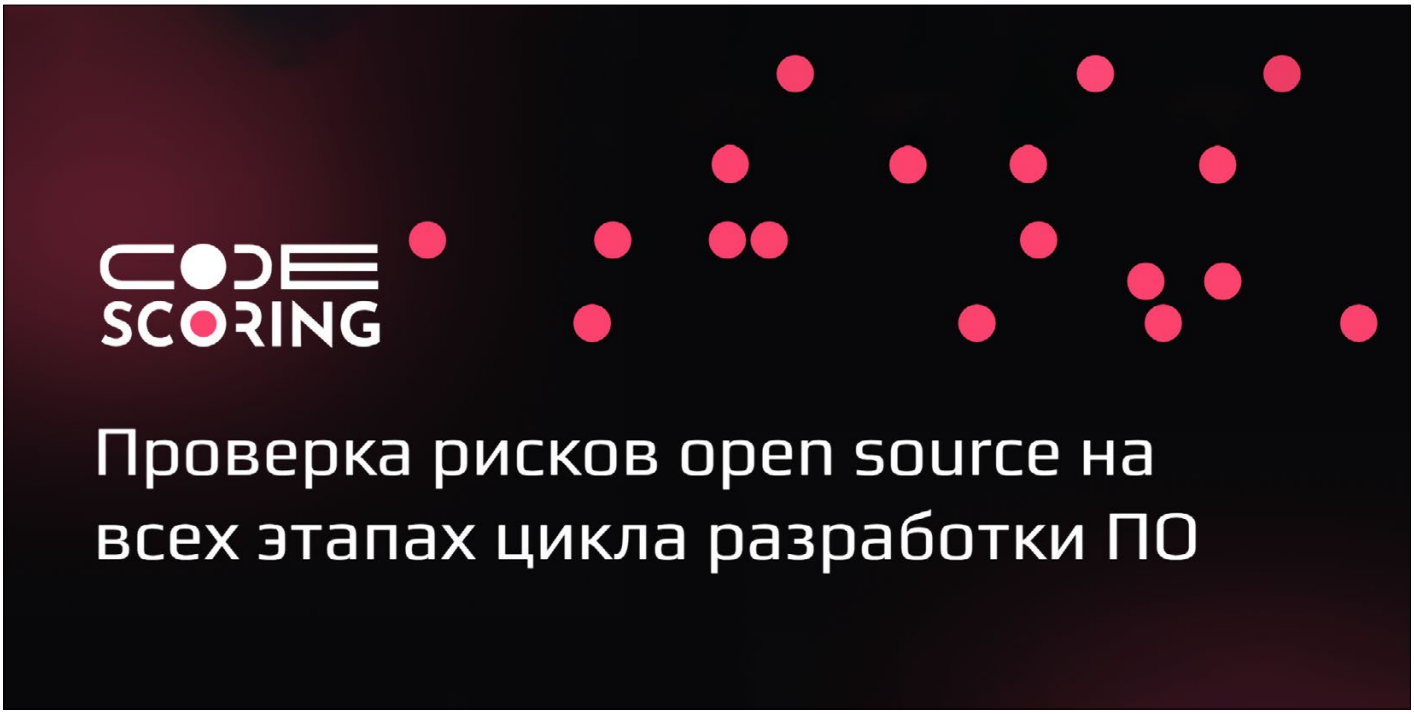
Для успешной реализации процессов разработки безопасного программного обеспечения в соответствии с ГОСТ Р 56939–2024 организациям необходимо сформировать команду, включающую различные роли, способные эффективно выполнять свои функции на каждом этапе жизненного цикла программного обеспечения. В процессе построения конвейера безопасной разработки крайне важно соблюдать принцип разделения полномочий и чётко определить для каждого члена команды конкретные элементы конфигурации, находящиеся в их зоне ответственности. Количество специалистов и их роли, представленные в таблице далее по тексту, являются усреднёнными значениями для компании и могут варьироваться в зависимости от специфики организации, масштаба разрабатываемого продукта и прочих факторов (таблица 3).

ВЫВОД

В статье рассматриваются особенности построения контура разработки безопасного программного обеспечения, который включает в себя интеграцию процессов, направленных на обеспечение безопасности, внедрение инструментария, применяемого в процессе разработки, а также вовлечение в выстроенные процессы команд, необходимых на каждом этапе жизненного цикла программного обеспечения. Подобный системный подход к организации всех указанных элементов является ключевым фактором для достижения высокого уровня безопасности программного обеспечения в организации.

Этап жизненного цикла ПО	Роль	Количество специалистов	Процессы из ГОСТ Р 56939–2024
Анализ требований	Product Owner	1 человек + заместитель	5.3
	Аналитик	2–3 человека	
Планирование	Руководитель проекта	1–2 человека	5.1, 5.2
	SCRUM-Master	2–3 человека	
Проектирование и дизайн	Архитектор	2–4 человека	5.6
	Архитектор по безопасности	2–3 человека	5.6, 5.7
Разработка ПО	Team leader	Для каждой команды разработки 1 человек	5.5, 5.8, 5.9, 5.10, 5.12
	Команда разработки	Различные команды от 5 человек	5.4, 5.5, 5.8, 5.9, 5.10, 5.16
	DevOps-специалист	На каждую команду разработки 1–2 человека	5.4, 5.5, 5.13
	Специалист по безопасности	На каждую команду разработки 1–2 человека	5.14, 5.15, 5.17, 5.18, 5.19, 5.24
Тестирование	Руководитель тестирования	Для каждой команды тестирования 1 человек	5.11, 5.18, 5.19, 5.24
	Команда тестирования	На каждый вид тестирования 3–5 человек	5.11, 5.18, 5.19, 5.24
Развёртывание	Специалист функционального сопровождения	2–3 человека на каждый продукт	5.20, 5.21, 5.25
	Специалист отдела внедрения и сопровождения (технической поддержки)	3–4 человека на каждый продукт	5.22, 5.23
	Инженер по сборке	2–3 человека на каждый продукт	5.4, 5.5, 5.13
	Инженер инфраструктуры	2–3 человека на каждый продукт	5.4, 5.5, 5.14
	Технический писатель	1–2 человека	5.4
Эксплуатация	Специалист по технической поддержке	3–5 человека на каждый продукт	5.22, 5.23
	Специалист по мониторингу и управлению инцидентами	1–2 человека на каждый продукт	5.23, 5.24

Таблица 3. Кадровый подбор для реализации требований ГОСТ Р 56939–2024



OPEN SOURCE: БЕЗОПАСНОЕ ИСПОЛЬЗОВАНИЕ

О БАЗОВЫХ ПРАКТИКАХ ЗАЩИТЫ ЦЕПОЧКИ ПОСТАВКИ, КОМПОЗИЦИОННОМ АНАЛИЗЕ ПО И ПОЧЕМУ ЭТО ПОЛЕЗНО ДЛЯ РАЗРАБОТКИ

Алексей СМИРНОВ
основатель CodeScoring

Обновлённый стандарт безопасной разработки (ГОСТ Р 56939–2024) определил практики, которые не были отражены в прежней версии стандарта. В частности, в отношении работы со сторонними компонентами описываются подходы к защите цепочки поставки и необходимости применения композиционного анализа ПО.

Контроль сторонних компонентов играет важную роль с точки зрения безопасности, поскольку важно следить за тем, что включается в состав вашего ПО. В статье рассматривается проблематика и действенные подходы, которые позволяют контролировать использование сторонних компонентов ПО от стадии выбора компонентов до особенностей их применения в продуктивных средах.

ПРОБЛЕМАТИКА

Открытого кода в программных продуктах становится всё больше, а культура ответственного потребления сторонних компонентов, к сожалению, отстаёт. Бизнесу нужно быстрее разрабатывать, а с техническим долгом и особенностями библиотек «мы разберёмся позже». Вместе с открытым кодом в ПО попадают недостатки, связанные не только с базовым качеством, но и с безопасностью. Важно помнить, что открытый код известен, исследуем и идентифицируем в составе ПО, которое может попасть в прицел злоумышленника. Кроме того, открытые компоненты обладают своими лицензионными соглашениями, ко-

торые читают, увы, не все пользователи, что может привести к последующей переработке кода и замене используемой библиотеки.

Помимо базовых вопросов безопасности и качества компонентов, в их отношении нарастает количество атак на цепочку поставки. Подобное происходит, когда злоумышленник намеренно выпускает версию пакета с полезной нагрузкой, что приводит к утечке параметров окружения (в частности, компрометации конвейера сборки), появлению бэкдоров в продуктах, занесению шифровальщиков в контур и иным последствиям.

Самые популярные способы подобной «доставки» — это:

- typosquatting — публикация компонента с заведомой опечаткой в названии;
- dependency confusion — эксплуатация механизмов управления компонентами;
- malicious code injection — инъекция вредоносного кода в тело легитимного пакета.

Кроме того, в последние годы обострилась ситуация с закладками и политизированным саботажем авторов компонентов.

СТАТИСТИКА

Команда CodeScoring обладает собственной экспертностью в части анализа данных о мировом Open Source, чем регулярно делится с сообществом РБПО. Приведём сокращённую статистику:

- ~100 уязвимостей и вредоносных активностей в компонентах обнаруживается ежедневно;
- ~900 вредоносных пакетов выявляется ежемесячно;
- 8 млн — мировая база открытых компонентов;
- 90 млн разработчиков участвуют в открытых проектах;
- ~2500 типов открытых лицензий со своими видами ограничений и разрешений.

ХРАНЕНИЕ КОМПОНЕНТОВ В КОНТУРЕ ОРГАНИЗАЦИИ

Если вы не храните компоненты, из которых производится или производилась сборка, пора начать делать это.

Для этого существует отдельный класс решений в виде **репозиториев артефактов**, которые позволяют не только хранить компоненты, но и интегрировать их в процессы сборки ПО с учётом модели разграничения доступа организации.

Одним из ключевых примеров здесь является открытый Nexus Repository OSS.

Теперь, если компонент пропадёт из интернета, вы всё равно сможете произвести сборку.

КОНТРОЛИРУЕМЫЙ РЕПОЗИТОРИЙ АРТЕФАКТОВ

Что касается безопасности компонентов, важно не только хранить их у себя, но и проверять на вредонос, публичные уязвимости, разрешённые лицензии и иные особенности компонентов, которые допущены к использованию внутри организации.

Такую проверку можно организовать для каждого запроса компонента пользователем.

Например, наше решение для защиты цепочки поставки CodeScoring. OSA позволяет встроиться в процесс запроса компонента разработчиком и произвести проверку на соответствие политике безопасности организации. В случае если компонент нарушает принятые политики, он блокируется, а разработчику даются подробные пояснения, почему так вышло и что делать дальше.

Узнать больше про защиту цепочки поставки с CodeScoring.OSA

https://codescoring.ru/osa

КОМПОЗИЦИОННЫЙ АНАЛИЗ, ИЛИ «ИЗ ЧЕГО СОСТОИТ ВАШЕ ПО»

Понятие композиционного анализа на отечественных SDL-пространствах является довольно молодым, но важным и встаёт в один ряд со статическим и динамическим анализами.

Если коротко, то этот вид анализа позволяет ответить на вопрос: «из чего состоит ваш продукт?» и проверить риски.

Если более развёрнуто, то композиционный анализ основан на инвентаризации сторонних компонентов ПО, определении особенностей их использования, составлении перечня известных уязвимостей и/или иных недостатков компонентов.

В общей практике применения композиционный анализ разделяется на три части:

1. Инвентаризация всех компонентов в составе ПО. Это крайне важная стадия, которая должна быть выполнена точным образом, с учётом особенностей применяемых языков программирования и окружений сборки. Для ПО на разных языках могут быть свои особенности определения компонентов, например для C/C++.

2. Анализ компонентов
Обогащение информации о компонентах данными об уязвимостях и лицензиях и иными известными факторами компонента: возраст, авторы, количество версий и пр.

3.Рекомендация и действие
После получения результатов анализа производится проверка политик безопасности и может быть выполнено действие: например, заблокировать сборку. Результаты проверки политик должны сопровождаться информацией о выявленных проблемах и предлагать решение, например безопасную версию компонента.

Важным артефактом реализации этапа инвентаризации и анализа является перечень программных компонентов (ППК или SBoM, Software Bill of Materials). Это машиночитаемый документ, который обеспечивает фиксацию результатов в едином формате. Им можно поделиться с коллегами или заказчиком в рамках поставки. Существует два основных формата

обмена данными: CycloneDX (OWASP) и SPDX (Linux Foundation). В России больше распространён первый.

Полезный аспект реализации практики композиционного анализа — рекуррентность процесса. В уже выпущенных компонентах уязвимости не появляются, уязвимости обнаруживаются.

За год в продукте без обновлений накапливается ~100 уязвимостей от сторонних компонентов

Это означает, что в отношении ранее выпущенного ПО нужно не только регулярно проводить композиционный анализ в целях выявления новых проблем, но также следует наладить процесс обновлений своего ПО так, чтобы иметь возможность выпустить безопасную сборку при обнаружении рисков.

Узнать больше про композиционный анализ с CodeScoring.SCA

https://codescoring.ru/sca

ПОЛЬЗА ПРИМЕНЕНИЯ КОМПОЗИЦИОННОГО АНАЛИЗА ДЛЯ РАЗРАБОТЧИКОВ

Применение практики композиционного анализа позволяет разработке комплексно следить за компонентами, которые заносятся в разрабатываемые продукты.

Своевременная актуализация компонентного состава позволяет иметь меньше проблем с обратной совместимостью со старыми версиями компонентов.

Опыт работы с известными уязвимостями в сторонних компонентах повышает уровень знаний разработчиков в области безопасности кода и позволяет быть более подготовленными к практикам статического и динамического анализа.

НА ЧТО ОБРАТИТЬ ВНИМАНИЕ ПРИ ВЫБОРЕ ИНСТРУМЕНТА КОМПОЗИЦИОННОГО АНАЛИЗА

Наличие своего агента для инвентаризации компонентов. Наличие своего агента для инвентаризации компонентов, как правило, даёт более полноценную, глубокую, комплексную интеграцию со всем решением. Агент важен для встраивания в систему сборки для получения более точного результата анализа. Использование открытых инструментов, таких как Trivy, cdxgen или dep-scan, накладывает ограничения: вендору приходится учитывать их особенности и зависеть от направления развития этих проектов. Наличие функций, нацеленных на сокращение издержек по работе с инструментом: точность анализа, гибкие политики безопасности, удобство работы с инвентарём и качественные рекомендации. Встраиваемость в действующую автоматизацию разработки. Наличие агента, программных интерфейсов, вебхуков существенно упростит.

ЗАКЛЮЧЕНИЕ

Наладив правильные процессы проверки сторонних компонентов для своих продуктов, вы получите основу и фундамент для построения безопасных процессов разработки:

- защиту цепочки поставки стороннего ПО;
- безопасность компонентной базы в первом приближении;
- информированность об уязвимостях в ранее выпущенных версиях ПО;
- понимание и контроль лицензионной чистоты;
- контроль актуальности компонентной базы выпускаемого ПО.

СПРАВКА О CODESCORING

Платформа CodeScoring представлена на рынке с начала 2021 года и сегодня решает задачи банков, телеком-операторов, ИТ-компаний и других организаций, для которых важны вопросы кибербезопасности и качества собственных продуктов.

CodeScoring сегодня:

- вся ключевая функциональность собственной разработки;
- анализ 15 языков программирования и Docker-образов;
- более 20 интегрированных баз уязвимостей;
- собственная база знаний про мировой open source;
- интеграция на всех этапах разработки ПО;
- широкий набор политик отслеживания и блокирования рисков;
- интеграция с отечественными ASOC/ASPM-платформами;
- интеграция с Kaspersky Open Source Software Threats Data Feed;
- применение машинного обучения для удобства пользователей.

Мы остаёмся одними из немногих производителей средств безопасной разработки, кто ведёт открытый Changelog и доступную документацию.

КОД БЕЗ СЕКРЕТОВ: ПОЧЕМУ ЭТО ВАЖНО?

К ЧЕМУ ПРИВОДИТ НАЛИЧИЕ КОНФИДЕНЦИАЛЬНОЙ ИНФОРМАЦИИ В КОДЕ ПРОДУКТОВ И КАК С ЭТИМ БЫТЬ

Алексей СМИРНОВ
основатель CodeScoring

ЧТО ТАКОЕ СЕКРЕТ?

ГОСТ Р 56939–2024 под секретами определяет «данные, которые могут использоваться для обеспечения аутентификации и/или целостности и/или конфиденциальности информации (пароли, цифровые сертификаты и т.п.)».

Секреты могут быть представлены также и иными данными, например платёжными: номерами карт, счетов, пинами и пр. Подобную «живую» информацию хранить в коде, конфигурационных файлах или фикстурах совершенно нежелательно.

ПОЧЕМУ ЭТО ВАЖНО

Исходные коды приложений находятся под стражей специалистов по безопасности. Тем не менее приложения, собранные из этих кодов, регулярно разворачиваются на сторонние площадки или публикуются (очевидный пример — мобильные приложения), а сами исходные коды хранятся на рабочей технике разработчиков. Угроза внутреннего злоумышленника и утечки исходных кодов сохраняется актуальной для любой организации.

Наличие чувствительной информации в приложении может позволить злоумышленнику использовать её для получения доступов в целях кражи персональных данных, поддержания цепочки взломов, фишинга, распространения вредоносного ПО или иных целей.

Одним из примеров реализации подобного вида атаки является декомпиляция мобильных приложений из публичных площадок с целью извлечения доступов к его публикации, размещённых разработчиками по неосторожности, что позволяет подменить оригинальное приложение приложением с полезной нагрузкой. Аналогичным образом получаются доступы к API используемых сервисов, sms-шлюзам, ключам доступа к платёжным сервисам, которые случайно забыла разработка.

СТАТИСТИКА

Показательной является статистика открытого кода из свежего отчёта The State of Secrets Sprawl 2024:

- 12,8 млн новых секретов обнаружено в публичных коммитах на GitHub за 2023 год;
- 90 % секретов оставались валидными и через 5 дней после утечки;
- каждый десятый разработчик случайно добавляет секрет в код.

К сожалению, ревью кода и средства идентификации секретов в коде до публикации изменений применяются не всеми и совершить подобную ошибку случайного добавления может каждый.

ПОСТРОЕНИЕ БАЗОВЫХ ПРОЦЕССОВ РАБОТЫ С СЕКРЕТАМИ

Важно контролировать, чтобы секреты не попадали в исходные коды приложений, хранились и обрабатывались соответствующим образом.

Во-первых, должно быть организовано пространство хранения и предоставления секретов смежным приложениям. Для этого есть специальные системы управления секретами («паролешницы»), например открытый Vault.

Во-вторых, следует настроить блокирующую проверку секретов на стороне разработки. Это могут быть такие открытые и «лёгкие» статические анализаторы, как GitLeaks, TruffleHog, DeepSecrets или иные, которые уже устоялись в экспертной практике и позволяют легко настраивать правила анализа. Независимо от уровня применяемого инструмента следует приготовиться к разбору ложноположительных срабатываний, если не предлагается специальных механизмов интеллектуального анализа. К примеру, наша лаборатория показала, что можно натренировать модель машинного обучения, которая с ходу идентифицирует 50–70 % ложноположительных срабатываний. А если проводить дообучение этой модели на контекстных данных проектов, то результат стремится к 90–95 %, что существенно экономит время специалистов РБПО на разбор срабатываний.

В-третьих, если всё-таки утекло или возникли подозрения, то должна быть разработана чёткая процедура отзыва и инвалидации утёкших секретов. Не говоря уже о том, что секреты должны ротироваться и вообще подчиняться определённой ролевой модели использования.

Безусловно, всё, что вы придумаете и настроите, должно быть закреплено в регламенте по работе с секретами. Или наоборот, но регламент должен работать, а не лежать.

ВЫВОДЫ

Можно разработчику сильно не доказывать, что секреты в коде — это неправильная инженерная культура. Ошибки совершают все. Но правильный процесс и инструментарий для контроля секретов должны быть реализованы.

БЕЗОПАСНОСТЬ — ЭТО ВАША ПРИБЫЛЬ

КАК УПРАВЛЕНИЕ БЕЗОПАСНОЙ РАЗРАБОТКОЙ НА ОСНОВЕ ДАННЫХ СОКРАЩАЕТ TIME-TO-MARKET

Антон БАШАРИН
старший управляющий партнёр AppSec Solutions

Разработка программного обеспечения, в том числе мобильных приложений, давно стала необходимой частью бизнес-процессов как в b2c-, так и в b2b-сегменте — в особенности это касается крупного и среднего бизнеса. В то же время некоторые организации относятся к вопросам кибербезопасности, включая разработку безопасного ПО, по остаточному принципу. В результате общий ущерб экономике России от действий злоумышленников в 2023–2024 годах, по оценкам Сбербанка, может превысить 1 триллион рублей, а в открытом доступе находятся данные 90% взрослых россиян. На этом фоне государство заметно и обоснованно будет ужесточать санкции за уязвимости в ПО, а клиенты — выбирать более безопасные сервисы.

Таким образом, разработка безопасного программного обеспечения, или DevSecOps, — это не затраты, а инвестиции. Сделать их максимально эффективными помогает использование методов управления Data Driven и принципа Shift Left. Объясним, что это и как работает.

ЗАЧЕМ НУЖЕН DEVSECOPS БИЗНЕСУ

Для начала разберёмся, как работает неэффективное управление процессами разработки и во что оно обходится бизнесу. Для этого очень хорошо подходит метрика Time-to-Market, или скорость вывода продукта на рынок — до конечного пользователя. Другими словами, чем раньше будет релиз, тем быстрее мы начнём зарабатывать. Поскольку прибыль от выведенного на рынок продукта остаётся примерно одинаковой, любые задержки разработки увеличивают затраты, что снижает рентабельность. Ошибки, выявленные на поздних этапах создания ПО, требуют больше ресурсов на исправление, поскольку при этом задействуются высокооплачиваемые специалисты: разработчики, системные архитекторы, техлиды и другие. Это особенно критично для уязвимостей информационной безопасности, поиск которых требует привлечения ИБ-инженеров или подрядчиков, что само по себе связано с крупными финансовыми вложениями. А устранение таких уязвимостей на поздних этапах значительно увеличивает трудозатраты, делая раннее их выявление и предотвращение ключевым фактором эффективности процесса.

Что означает доработка на позднем этапе? Это аврал, отвлечение ключевых специалистов от других задач и проектов, сверхурочная работа. Представьте: у вас всего два ИБ-инженера, а тестирование выявило 500 серьёзных уязвимостей. Очевидно, что они физически не смогут справиться с таким объёмом за две недели, даже если предложить им дополнительные выплаты. Привлечение ещё 10 специалистов практически нереально, так как это сверхдефицитная специальность. В результате сроки проекта неизбежно сдвинутся на неопределённое время, затраты станут неконтролируемыми, и возникнет реальный риск того, что проект в итоге окажется нерентабельным.

К сожалению, такой результат возможен даже в том случае, если в компании применяются отдельные практики разработки безопасного программного обеспечения, например, в силу требований регулятора или если замруководителя по ИТ в целом фокусируется на информационной безопасности. Какие-то регламенты принимаются, те же сканеры закупаются. Но если средний менеджмент и сами разработчики относятся к требованиям безопасности как к навязанным извне задачам, которые нужно выполнить «для галочки», провалы неизбежны.

Именно поэтому важно внедрять принципы DevSecOps с акцентом на подход Shift Left. Этот подход предполагает смещение практик обеспечения безопасности влево по таймлайну проекта — с этапа приёмки на этапы разработки или даже проектирования. Это критично, поскольку именно на ранних этапах создание и обнаружение новых рисков информационной безопасности наиболее вероятно, а их устранение обходится значительно дешевле, чем на более поздних стадиях.

На ранних этапах разработки нужно внедрять сканеры, которые сразу анализируют уязвимости в исходном коде. Они же становятся фактическим барьером для попадания вредоносных элементов при использовании источников Open Source. На следующих этапах применяется уже динамический анализ. Например, сканируется API, то есть программный (API) и пользовательский (UI) интерфейс.

То есть анализ идёт на каждом этапе разработки, и в финальной сбор-

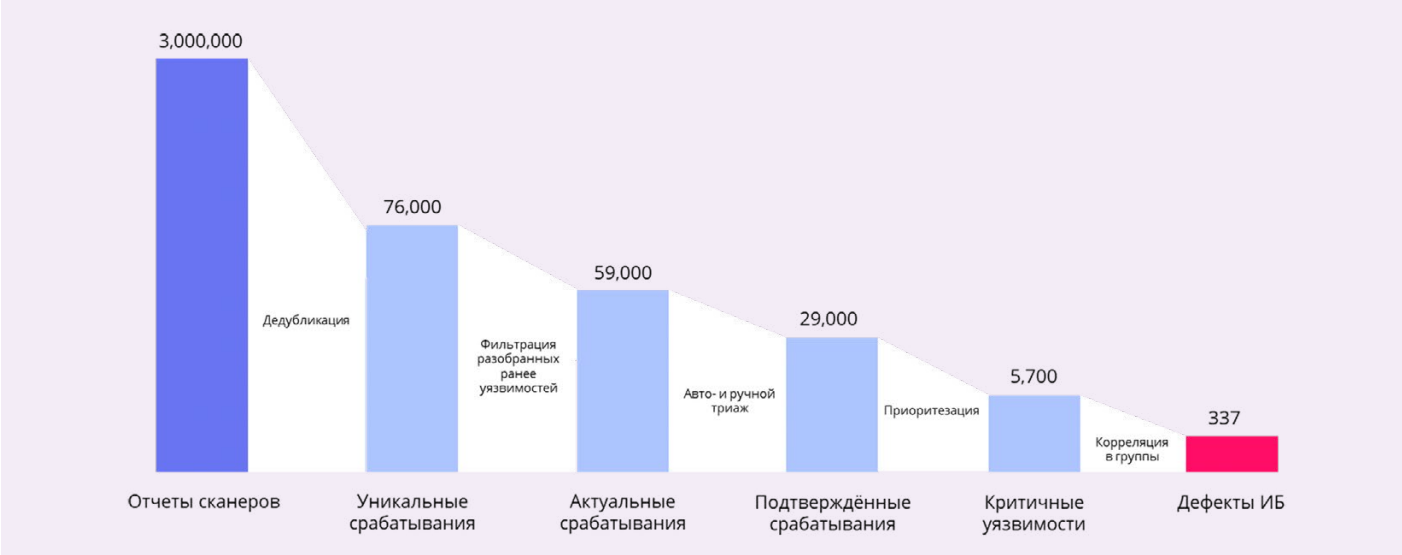


Рисунок 1. Эффективность обработки результатов сканирования ASPM-платформы AppSec.Hub

ке могут остаться единицы уязвимостей, а не сотни, как могло бы произойти без применения принципов DevSecOps.

Организация комплексного контроля и аналитики безопасной разработки может существенно ускорить и упростить создание программных решений. Она позволяет сформировать своего рода единый центр управления сканированием уязвимостей, оценкой рисков и анализом состояния кода.

ДЛЯ ЧЕГО НУЖНЫ ASPM-РЕШЕНИЯ

ASPM (Application Security Posture Management) — это класс систем для управления безопасностью приложений, который помогает организациям оценивать, управлять и улучшать свою безопасность приложений на всех этапах жизненного цикла разработки. Платформа ASPM предоставляет инструменты и средства для анализа уязвимостей, контроля за соблюдением стандартов безопасности, управления инцидентами и выполнения аудитов. Основные функции ASPM-решения:

1. Оценка и приоритизация уязвимостей. Помощь в выявлении слабых мест в приложениях, включая сортировку уязвимостей, оценку всех данных со сканеров.

2. Контроль за соблюдением стандартов. Проверка соответствия приложений положениям нормативных актов и корпоративным стандартам кибербезопасности.

3. Управление рисками. Помощь в оценке рисков, связанных с уязвимостями и угрозами, для более осмысленных решений в области безопасности.

4. Определение корреляций и аналитика. Визуализация данных по уязвимостям и контроль процессов РБПО на основе прозрачной системы метрик.

Давайте разберём, из чего состоит типовой процесс управления безопасностью приложений с помощью ASPM. Пройдя через все стадии автоматической и ручной обработки, анализа, приоритизации и корреляции, уязвимости преобразуются в дефекты, экспортируемые в дефект-трекинговую систему команды разработки для исправления.

Основные стадии обработки результатов сканирования:

- дедубликация,
- применение автоматических правил обработки,
- триаж,
- приоритизация,
- корреляция.

Примером такой системы является российская платформа AppSec.Hub (рис. 1). Это централизованная платформа для интеграции различных инструментов безопасности и процессов. Основные преимущества AppSec.Hub:

1. Единая точка управления. AppSec.Hub обеспечивает централизованное управление всеми инструментами безопасности, что упрощает процесс мониторинга и обеспечения безопасности приложений.

2. Интеграция инструментов. Позволяет объединять данные из различных источников и инструментов, используемых для анализа безопасности приложений, и предоставляет полную картину безопасности.

3. Аналитика и отчётность. Упрощает процесс анализа уязвимостей и генерирует отчёты, которые могут быть использованы для принятия решений и управления рисками.

4. Прозрачность и контроль на основе метрик. Предоставляет командам разработки и безопасности возможность видеть состояние безопасности всех приложений и управлять им из одного интерфейса.

Особо отмечу, что все рутинные операции автоматизируются в том числе с применением искусственного интеллекта, как, например, в нашем продукте AppSec.Hub с помощью модуля AppSec.CoPilot. Это позволяет отсеивать ложные срабатывания — как правило, это более 90 % того, что выдают сканеры. Время ваших специалистов в таком случае тратится на решение действительно сложных и необходи-

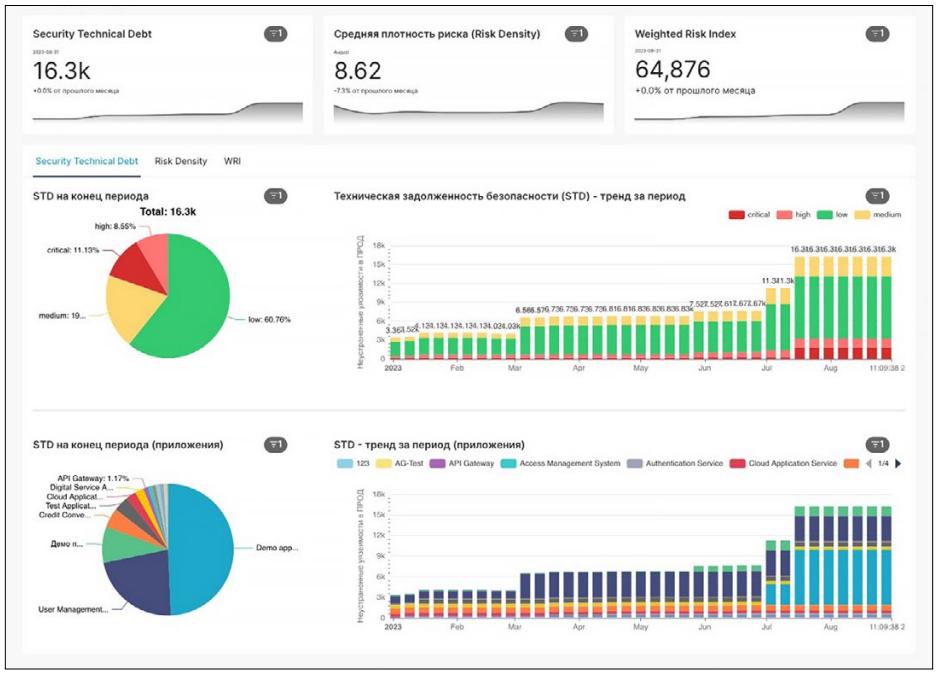


Рисунок 2. Дашборд операционных показателей ASPM-платформы AppSec.Hub

мых задач, а алгоритмы ИИ не требуют почасовой оплаты.

Принцип Shift Left и автоматизация делают методологию DevSecOps максимально эффективной. В идеале все инструменты ИБ интегрируются в среду разработки, проще говоря, на рабочий стол специалиста. В реальности проверки ИБ встраиваются в CI/CD- или DevOps-процесс, который запускается автоматически после публикации изменений разработчиком. Одновременно формируется аналитика, представление которой должно быть адаптировано под уровень получателя. Например, руководитель проекта должен видеть подробную динамику выявленных и закрытых уязвимостей. А вице-президенту по ИТ или гендиректору достаточно светофора: если процессы проекта идут в «зелёной зоне», ему не нужно вмешиваться, а если попадают в «жёлтую» или вдруг включается «красный», то нужно тормозить и решать выявленные проблемы на данном этапе, а не в конце, потому что потом это обойдётся намного дороже.

DevSecOps — это не набор определённых инструментов и правил их использования, а именно бизнес-подход.

Представим основные метрики аналитики уязвимостей в виде пирамиды по уровням принятия решений в бизнесе от инженерных команд к ценности продукта.

◆ На её острие будет уровень топ-менеджмента (Executive). С точки зрения руководителя компании он влияет на следующие ключевые метрики: Time-to-Market (о чём мы уже говорили подробно), Digital Business Continuity & Security (защищённость и непрерывность цифрового бизнеса), Compliance (соответствие требованиям регулятора), Software Development Maturity (зрелость процесса разработки). Все эти параметры составляют общую ценность от внедрения безопасной разработки.

◆ «Вторым слоем» пирамиды мы представляем себе управленческий уровень (Management). На нём расположены метрики, на которые влияет управленческая команда проекта. Например, менеджмент контролирует time-to-market для определённого релиза, покрытие продуктовых команд практиками Application Security, устранение уязвимостей в программном продукте и т.д.

◆ Операционный уровень (Operations) — информационный слой, являющийся производным от параметров телеметрии. В зоне ответственности операционного уровня такие метрики, как объём технического долга, плотность дефектов, среднее время жизни дефекта и т.д. Параметры этого уровня, как правило, контролируются на уровне отдельных команд (рис. 2).

◆ База пирамиды — уровень телеметрии (Telemetry) — это вся информация, которая поступает от сканеров из DevOps- или CI/CD-процессов, из репозиториев исходного кода и артефактов сборки. Это гигантский объём данных, с которым работает команда. Но все они нуждаются в структуре и приоритизации — без этого невозможно подняться на уровень выше. Мы собираем информацию в специально разработанном и структурированном хранилище. Интерпретация и представление данных в виде удобных и понятных диаграмм на дашбордах для разных уровней управления — это фишка AppSec.Hub. Этот продукт позволяет реализовать полноценный подход Data Driven к управлению DevSecOps-процессом, эффективность которого будет только расти по мере усиления роли и функционала AI-инструментов.

* * *

Разработка безопасного программного обеспечения является неотъемлемой частью современного бизнеса, независимо от его масштабов и сферы деятельности. Анализ ситуации на рынке показывает, что игнорирование вопросов информационной безопасности может привести к колоссальным экономическим потерям и утечкам данных, что ставит под угрозу не только бизнес, но и доверие клиентов. Применение принципов DevSecOps и Shift Left в процессах разработки закономерно приводит к снижению рисков и затрат на исправление уязвимостей. При этом ASPM-решения становятся гибкими инструментами, которые помогают организациям измерять, управлять и улучшать безопасность приложений в течение всего жизненного цикла разработки.

ЭКСПЕРТОВ С НИЗКОЙ ОБРАЗОВАТЕЛЬНОЙ БАЗОЙ ЗАМЕНИТ ИИ

НО СПРОС НА ТАЛАНТЫ БУДЕТ ТОЛЬКО РАСТИ

Дарья ФИГУРКИНА
директор по талантам Swordfish Security

1 миллион человек — примерно столько профессионалов не хватает ИТ-отрасли России. В кибербезопасности этот дефицит ещё острее. Причин этому две. Во-первых, технологии развиваются быстрее образовательных программ, во-вторых, нет доступа к рынку ИТ-кочевников, которые находятся за пределами страны. Кибербезопасность, пожалуй, самая чувствительная отрасль в сфере разработки, поскольку её основная задача — обеспечить цифровую защищённость критической инфраструктуры и коммерческих компаний внутри России.

Дефицит кадров в отрасли будет только усугубляться. Число вакансий для узких специалистов, например для редчайших ML-инженеров, за последний год стало больше на 53% (по данным hh.ru — *прим. автора*), а зарплатное предложение для ИБ-инженеров за год подросло на 1/5. Сложное время для HR, зато потрясающее — для молодых и талантливых инженеров, которые только входят в профессию, ведь для них сегодня открыты все двери, в них готовы инвестировать работодатели и доращивать их внутри своих команд. Пожалуй, DevSecOps наименее требовательна к базовому образованию за счёт высокой мобильности внутри профессии.

У нас в компании работает определённое количество студентов 3–4-го курса. Молодые люди начинают карьеру без отрыва от учёбы на 0,5 ставки, за год или два вырастая в высококлассных инженеров. В прошлом году мы наняли человека, который окончил первый курс, но он гений и уже работает на равных. Взять и выучить под себя — это очень логично, сегодня студенты технических вузов не борются за место на рынке труда, потому что это рынок борется за них. Глобально у этого тренда есть серьёзный минус: в долгосрочной перспективе такой быстрый взлёт может привести к просадке качества работы в ИБ.

Впрочем, мы находимся, на мой взгляд, на пороге очередной цифровой революции, которая всё изменит. Драматично рынок DevSecOps, да и ИТ в целом трансформирует массовое внедрение генеративного искусственного интеллекта, который возьмёт на себя простые задачи, которые можно чётко сформулировать. Это сильно скорректирует рынок труда в нашей сфере. Просто хороший код будет писать модель. Приведу простой практический пример: уже сейчас наши коллеги используют модель AppSec.CoPilot на основе ИИ, чтобы в автоматическом режиме анализировать и сортировать уязвимости, определять ложные срабатывания. Практика показала, что его точность 96%. Таким образом получается сэкономить рабочее время команды из 5 инженеров, которые потратили бы на этот процесс 8000 часов в год.

Человеческие ресурсы будут востребованы для более творческих задач. Чтобы оставаться ценным, ты должен уметь создавать. Техническую революцию успешно переживут талантливые инженеры, которые готовы развиваться, способные на глобальное мышление. Самым ценным станет то самое helicopter view, которое достаточно сложно в себе развить, не имея хорошего фундаментального образования. Которое, к моему сожалению, не успевают получить быстро развивающиеся в профессии молодые инженеры, а вузы не успевают трансформировать программы под современность.

Ещё один тренд: рынок соискателя, а в ИТ-отрасли сегодня именно он, предполагает, что выбор условий остаётся на стороне работника. Примером служит удалёнка, которая прочно вошла в нашу жизнь в пандемию и из жизни ИТ никуда больше не ушла. От 60 до 70% сотрудников, если мы говорим об инженерах, остаются на удалёнке. Это открывает возможности для подбора, потому что ты не привязан к поиску в Москве, талантливые инженеры могут работать с кодом в Красноярске, Калининграде и Мурманске. Мой личный топ (кроме очевидных МГТУ им. Баумана и МФТИ) возглавляет Сибирь, поскольку там есть вузы, которые дают качественное фундаментальное техническое образование. Это расширяет возможности для найма, поскольку удалёнка из тренда переросла в норму.

Ещё один тренд — более открытая индустриальная среда. Работодателю приходится учитывать личные планы на профессиональное развитие членов команд и давать им свободу

творчества. Сегодня одно из обязательных условий привлечения хороших ИБ-инженеров — команда экспертов, в которой есть у кого и чему научиться, готовая поддерживать инициативу и воплощать новые идеи. Руководить директивно «сверху» уже не получится. Чтобы успешно работать на рынке кибербезопасности, важно умение объединять профессионалов вокруг общей идеи и быть открытыми к тестированию гипотез. Бонус — мы получаем среду, которая непрерывно улучшает сама себя.

Чтобы эта система состояла из звеньев, каждое из которых достаточно автономно, чтобы привнести нечто новое и полезное, но при этом заряжено общей целью, стандартный цикл сотрудника начинается с анализа предыдущего опыта кандидата. Глядя на резюме кандидата, даже если в нём нет пометки «ищет работу», можно определить траекторию его развития и понять, какой проект можно было бы ему предложить, что могло бы стать логичным этапом его развития, и сопоставить это с актуальным запросом компании. Таким образом, мы не просто нанимаем в команду единицу, «винтики», а находим того, с кем совпадают наши цели и ценности. А дальше наша задача — создать условия, чтобы таланты могли проявиться в полной мере.

Когда речь идёт о сравнительно небольшой команде экспертов в узком сегменте РБПО, можно позволить себе роскошь выбирать и растить. Но рынок, где требуются специалисты по кибербезопасности, гораздо больше. Они востребованы на производстве, в финансовых и ТЭК-компаниях, в сервисах электронной торговли, да и вообще везде. Это сотни тысяч сотрудников, которых сложно найти, ещё труднее привлечь и удержать. Кадровый голод частично решается за счёт аутстаффинга и построения центров компетенций в DevSecOps для коммерческих компаний. За последние три года спрос на эту услугу вырос многократно. Если говорить о Swordfish Security, то в семь раз. Мы начинали этим заниматься в 2022 году в рамках «Академии кибербезопасности», и если в первый год работы на счету команды было три готовых центра компетенций на стороне клиента, то в 2024-м их более 20, а количество заявок на порядок выше. К нам регулярно обращаются наши партнёры с запросом о подготовке лидеров команд кибербезопасности внутри компаний. Мы видим, что не только отдельные компании-эксперты, но и рынок в целом готов вкладывать в людей, которые смогут делиться своими знаниями о разработке безопасного ПО со своими коллегами и вдохновлять их на развитие.

* * *

Технологии развиваются так быстро, что просто переждать кадровый голод в кибербезопасности, перекрывая его исключительно разовыми аутсорсинговыми услугами, уже не получится. Культуру кибербезопасности предстоит развивать во всех отраслях экономики, а значит, и кадры для неё выращивать. Дорогу осилит идущий, а в пункт назначения первыми доберутся те, кто делает серьёзные шаги уже сейчас.

«ГОРЕСТИ И РАДОСТИ» ПРАКТИЧЕСКОГО РБПО

ВЗГЛЯД РБПО-СООБЩЕСТВА ФСТЭК РОССИИ И ИСП РАН

Дмитрий ПОНОМАРЁВ
НТЦ «Фобос-НТ» / ИСП РАН / МГТУ им. Баумана
Интересы: аудит и консалтинг построения процессов безопасной и качественной разработки ПО, анализ поверхности атаки

В этой подборке опытом «удивительных открытий» и решений практических задач в области безопасной и качественной разработки поделятся коллеги — известные в сообществе инженеры — рангом от руководителя направления до владельца компании. Мы подобрали палитру рассказчиков так, чтобы охватить максимум различных технологических профилей — чтобы и сами истории, и стилистика их рассказа оказались интересны и полезны широкому кругу специалистов.

К числу моих любимых историй относится опыт разбора поверхности атаки с одним из заявителей на сертификацию. В ходе первичного анализа программно-аппаратного решения мы обнаружили там полноценный Jabber-сервер, написанный на C++. На вопрос «Зачем вам это в сертифицируемом продукте в 2022 году?» последовал ответ: «Да просто N лет назад какой-то один клиент попросил включить «за копеечку», ну, мы и включили, а в чём дело?» Далее последовала моя минилекция о стоимости и целесообразности фаззинг-тестирования сетевого Stateful-протокола в рамках и так горящей сертификации, в ходе которой верное решение — «Доктор сказал: резать» — было принято через 15 минут. Мораль истории проста: убирайте из состава вашего продукта весь непрофильный «для бизнеса» код, особенно код сетевых сервисов, формирующих поверхность атаки. Особенно в случае получения вашего первого сертификата соответствия. Пара недель на рефакторинг смогут сэкономить вам месяцы работы и миллионы рублей трудозатрат, не говоря уже о повышении защищённости продукта.

Ещё одна забавная (и типичная для отрасли) история, непосредственным участником которой являлись я сам и наши друзья из «Айдеко», опубликована ранее и доступна по ссылке (см. врезку в разделе «Выводы») [https://www.itsec.ru/articles/kak-pravilno-organizovat-process-bezopasnoj-razrabotki-po-sdl-6-shagov].

Желаю всем приятного и интересного «РБПО-шного чтива»!

Дмитрий СОРОКИН
технический директор компании «Базис»

На этапе внедрения безопасной разработки в наши продукты и проведения первичного аудита безопасности и качества кода наша команда инженеров столкнулась с неожиданной находкой: анализ выявил критические уязвимости в 150 библиотеках одного продукта. А до окончания спринта и выхода в релиз оставалось всего две недели. После проведённого анализа потребовалось выяснить, откуда взялось такое количество уязвимостей и как их исправить в такой короткий срок. На проведённом митинге с командой разработки после долгого разговора выяснилось, что все библиотеки относились к части неиспользуемого, устаревшего кода, о котором все, естественно, благополучно позабыли, пользуясь правилом «работает — не трогай».

Таким образом, просто избавившись от ненужного придатка в продукте, команда доблестно выявила критические уязвимости и также не менее доблестно их закрыла одним движением руки. С того момента команда разработки не оставляет устаревший код, но команда инженеров по безопасной разработке пристально за этим следит. А с появлением в нашем арсенале инструментов, созданных инженерами ИСП РАН и CodeScoring, мы повысили контроль за качеством и безопасностью нашего кода и выпускаемых нами продуктов. Это, в свою очередь, гарантирует надёжность и безопасность продуктов компании «БАЗИС».

https://corp.cnews.ru/news/top/2025-01-24_rossiyane_ispravili_pochti

Николай КОСТРИГИН
руководитель отдела БРПО, компания «Базальт СПО»

Интересы: развитие свободного ПО в целом и повышение доверия к нему в частности

Согласно укоренившемуся в сообществе РБПО тезису «Заимствованный код — твой код!», команда «Базальт СПО» постоянно проводит исследования программных пакетов, входящих в репозиторий свободного ПО «Сизиф» и его стабильные бранчи. Работа ведётся как самостоятельно, так и в кооперации с другими компаниями-разработчиками. Используется широкий арсенал свободных и проприетарных инструментов.

Не так давно сотрудничество в рамках Центра исследований безопасности системного программного обеспечения привело к интересному случаю, существование которого предполагалось, но на практике мало кто встречал. При исследованиях кода Qemu фаззингом было выявлено падение, получившее оценку серьёзности 6.6 по CVSS 3.1. Разработанный патч готовился к отправке в апстрим Qemu, когда выяснилось, что такое же исправление уже подготовлено другой компанией — участником консорциума, но как результат анализа срабатывания статического анализатора. Получился своего рода «выстрел Робин Гуда».

Интересные случаи, связанные с безопасностью свободного ПО, встречались в проекте «Сизиф» и раньше. К примеру, в отношении CVE-2017–1000408 и CVE-2017–1000409. При анализе кода библиотеки GNU libc, используемой в операционных системах «Альт», на подверженность этим уязвимостям выяснилось, что они были устранены в ней превентивно ещё в 2001 году.

Вообще говоря, принимая участие в разработке и доработке проектов СПО, мы относимся к РБПО-исследованиям не как к неизбежной, навязанной кем-то обузе, но как к возможности погрузиться в исходный код проекта, узнать его глубже изнутри.

Андрей КУЗНЕЦОВ
руководитель Центра внедрения и развития практик РБПО НТЦ «Фобос-НТ»

Интересы: построение комплексных РБПО-процессов и последующая сертификация СЗИ

Современные испытательные лаборатории, желающие соответствовать актуальным отраслевым стандартам, требованиям и стремительно меняющейся нормативной документации, неминуемо превращаются в «настоящие ИТ-компании» со спринтами, беклогами и пэйплайном. Эксперты лаборатории из классических «бумажных безопасников», всё больше становятся похожи на настоящих инженеров-исследователей, а сами лаборатории сталкиваются с суровой реальностью жизненного цикла разработки: если процесс не автоматизирован, его системно не существует. Автоматизация процессов тестирования ПО имеет первостепенное значение, поскольку обеспечивает, среди прочего, доступность информации о результатах тестирования для всех заинтересованных сторон, а так же воспроизводимость его результатов, не говоря о кратном ускорении процесса испытаний.

Комплексный подход к тестированию разрабатываемого ПО — это то, чего требует регулятор от компаний-разработчиков средств защиты информации именно сейчас. В частности, внедрения в жизненный цикл разработки программного обеспечения практик статического и динамического анализа (в том числе фаззинг-тестирования), что позволяет находить дефекты кода на ранних стадиях разработки. А это, в свою очередь, способствует повышению надёжности и стабильности работы ПО, многократному снижению затрат на экстренные исправления проблем «ушедших в прод».

Когда-нибудь эксперта лаборатории заменит бездушный и беспристрастный робот, но пока этого не случилось, экспертам необходимо автоматизировать свою деятельность и самостоятельно вживлять себе «кибернетические руки». Одним из процессов, который нам приходится автоматизировать, является фаззинг-тестирование — полезная, но нетривиальная практика, особенно если её выполняют разрозненные группы инженеров на своих ноутбуках. При такой организации деятельности даже простой сбор артефактов превращается в настоящее испытание.

Несколько раз обжёгшись, мы решили организовать собственный конвейер фаззинг-тестирования, автоматизирующий этапы: подготовки набора входных данных; запуска реализованных фазз-тестов; заведения задач по исправлению дефектов кода. Заручившись поддержкой и вычислительными мощностями коллег из ОрлГУ, начали изобретать свою «кибернетическую руку», использующую более 400 процессорных ядер и 500 Гб оперативной памяти. Как результат, уже за первый месяц пилотной работы конвейера многократно увеличилось количество обнаруженных дефектов кода, а число направленных в апстрим патчей (в т.ч. устраняющих уязвимости) исчисляется десятками: *https://fobos-nt.ru/novyie-dostizheniya-v-staticheskom-i-dinamicheskom-analize.html*

Антон ГАВРИЛОВ
владелец продукта Axel PRO

Интересы: безопасная разработка / разработка безопасного ПО и всё, что с ней связано

Многие в разговорах про безопасность видят только «лишения» и «ограничения». Однако зачастую практики РБПО могут помочь и разработчикам.

Знание объекта защиты — одна из самых базовых и важных вещей в информационной безопасности. Так же и с разработкой безопасного ПО. Важно понимать, из чего оно состоит, и по возможности минимизировать количество «лишних» компонентов. При этом важно не забывать, что «чужого кода» не существует: всё, что есть в ПО, оно «ваше».

Мне довелось участвовать в процессе сертификации достаточно «массивного» ПО, в котором присутствует очень много различных сервисов, работающих вместе, как единый организм.

В процессе изучения пакетов и образов контейнеров мы с разработчиками выяснили, что:

- для сборки многих образов используются похожие базовые образы; именно что похожие: golang:1.20.7, golang:1.20, golang:1.20.6, golang:1.20.14 и т.д.;
- в дистрибутиве содержатся образы, которые более не используются для корректной работы ПО;
- в дистрибутиве содержатся пакеты и образы разных версий, некоторые из которых не используются вовсе.

Почему это так? Ответ, как и всегда: «Исторически так сложилось». Да, где-то «недосмотрели», где-то «забыли», где-то «не знали».

В той выборке, о которой я говорил, суммарно получилось около 3000 разных пакетов в базовых образах контейнеров. Да, уникальных было в разы меньше, но как всё это поддерживать и анализировать, непонятно.

Было принято решение «всё почистить» — по возможности привести все базовые образы к единому «знаменателю», удалить ненужное и оставить только то, что реально используется.

В итоге осталось несколько базовых образов (ввиду специфики их «содержания»), а количество пакетов в них — не более 250 штук. Аналогичная ситуация (пусть и с меньшим «разбросом») получилась и с пакетами дистрибутива.

Да, классическая «инвентаризация», с которой начинается (практически) любая книга по информационной безопасности. Такой подход позволяет не только повысить уровень ИБ за счёт «отсечения лишнего», но и помогает разработке: вместо того чтобы поддерживать n базовых образов, можно поддерживать один и быть уверенным в том, что именно передаётся конечному пользователю и с какой версией.

Анатолий КАРПЕНКО
инженер по автоматизации, Luntry

Интересы: программная инженерия, безопасность ПО во всех её аспектах, надёжность ПО

В любой момент, всегда есть уязвимости, вероятность наличия НДВ, нежелательной функциональности. Этого не нужно бояться, нужно принять как факт и уметь работать в такой ситуации. Один из примеров — случай с CNI Calico. Проблема заключалась в том, что по умолчанию он отправлял телеметрию из инфраструктуры клиента своим разработчикам. Это было обнаружено совершенно случайно.

В этом и подобных случаях достаточно поправить флаги конфигурации запуска, но есть ещё куча других менее известных и документированных проектов, которые затаскивают разработчики к себе. Как они ведут себя, не очень очевидно, и это всё может привести к компрометации данных клиента. Поэтому к любому коду, как своему, так и стороннему, надо относиться с недоверием и выстраивать ZeroTrust-инфраструктуру.

Одним из подходов является уменьшение поверхности атаки и следование принципу наименьших привилегий, например, с помощью контейнеров. Оркестратор контейнеризированных приложений (например, Kubernetes) позволяет не только применить харденинг для самого контейнера (запретить запуск недоверенных файлов, сделать файловую систему доступной только на чтение и ограничить права пользователя), но и сделать ограничения на уровне самого оркестратора (настроить сетевые политики, мандатный доступ и т.д.). Экосистема вокруг контейнеризации позволяет организовать детектирование подозрительной активности, реакцию на неё, а в некоторых случаях вообще предотвратить её появление.

А как насчёт парадокса: безопасного ПО нет, а безопасная разработка есть? Тут все процессы надо внедрять, так как синергия безопасной разработки с безопасной эксплуатацией даёт максимальный успех, ведь всё это делается не ради процесса, а ради конечного результата.

Всем РБПО!

Алексей СМИРНОВ
основатель
CodeScoring

Интересы: полезные инструменты анализа ПО

100 УЯЗВИМОСТЕЙ В ГОД

Мы у себя в CodeScoring постоянно собираем и проверяем информацию про мировой Open Source. Эти данные нужны для защиты цепочки поставки и повышения точности композиционного анализа ПО. Собираем данные про сами компоненты (библиотеки) и их свойства, например: информация про уязвимости, патчи, эксплойты, а также данные о версиях, авторах компонентов, датах релизов и пр. Эти данные мы обрабатываем и зачастую узнаём много интересного, чем было бы полезно поделиться с сообществом.

В частности, мы постоянно строим и анализируем SBoM от множества открытых проектов. SBoM — перечень компонентов ПО, в котором указываются их версии и дополнительная информация об уязвимостях и условиях использования (лицензиях). В России он называется ППК — перечень программных компонентов.

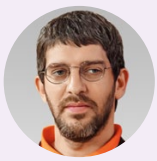
Однажды сформировав для своего ПО такой перечень, возможно проводить его актуализацию и следить за тем, какие уязвимости были обнаружены уже после того, как вы включили сторонний компонент к себе «под капот».

С одной стороны, формирование такого списка помогает более удобным образом следить за тем, сколько всего стороннего и с какими проблемами мы поставляем нашим заказчикам. С другой стороны, после выпуска даже самого «чистого» и безопасного продукта нет гарантии, что в сторонних компонентах со временем не обнаружатся новые проблемы. Поэтому ППК следует перестраивать (обогащать) и следить за новостями.

Наши последние расчёты показали, что в среднем ПО без обновлений за один год накапливает 100 (сто!) новых уязвимостей, из которых 10–15 являются эксплуатируемыми.

Отсюда следует логичный вывод, что, оценивая безопасность ПО, которое вы хотите использовать у себя, достаточно познакомиться с его динамикой релизного цикла. Если релизы раз в полгода, то есть над чем задуматься.

Следить за подобными проблемами помогает понятие композиционного анализа, которое было поставлено в один ряд со статикой и динамикой в обновлённом стандарте по безопасной разработке (ГОСТ Р 56939–2024). Верим, что осознанность использования сторонних компонентов и практики безопасной работы с ними будут только усиливаться.



Марк КОРЕНБЕРГ
технический директор
ООО «Айдеко»

Интересы: сноуборд, путешествия, инвестирование

В «Айдеко» мы ответственно подходим к кодовой базе, её минимизация и анализ занимают важную роль в процессах разработки безопасного ПО, и эта работа проводится нами самостоятельно, без давления со стороны регуляторов.

Так, во время сертификационных испытаний межсетевого экрана Ideco UTM специалисты «на спор» решили поискать интерпретируемый код и, что удивительно, нашли два избыточных интерпретатора, хотя главный архитектор был уверен, что ничего, кроме Python-кода, в составе дистрибутива нет. Ими оказались не используемый нигде Perl, который подтягивался из-за зависимостей, а ещё движок Mozjs из состава Firefox, который использовался для системы конфигурации очень популярного демона Policykit. В этом демоне конфигурация пишется на JS, а сам демон отвечает за систему раздачи прав пользователям и работает с root-привилегиями! В итоге Perl удалили из системы с помощью патчей компонентов, убирающих ненужную функциональность, и добавили проверку, чтобы он гарантированно не мог быть установлен. А PolicyKit переписали на Python, тем самым сократив кодовую базу. Такой подход позволил существенно упростить прохождение сертификационных испытаний.

Во время аудита и предварительной подготовки к сертификации процессов безопасной разработки мы нашли то, чего не должно быть в продукте (речь идёт о программе-агенте для рабочих станций), а именно библиотеки libGLX и libOpenGL. Это было похоже на настоящие археологические раскопки, и оказалось, что корень проблем — неверно собираемый фреймворк Qt, который по зависимостям притягивал много библиотек, без которых вполне можно обойтись.

Без использования OpenSource ПО написать современное решение невозможно. Но и с ними — опасно, поэтому регулярное устранение ненужной кодовой базы, обновление, различные виды анализа системы и кода, тщательный выбор компонентов и поставщиков ПО — это необходимость для безопасной разработки программного обеспечения в условиях увеличения функциональности продукта и роста команды разработки.

РБПО — МОДНЫЙ ТРЕНД ИЛИ НЕОБХОДИМОСТЬ?

ДИСКУССИИ НА ЭТУ ТЕМУ ПОДНИМАЮТ УРОВЕНЬ КУЛЬТУРЫ ИБ

Вячеслав ПОЛОВИНКО
руководитель направления собственных продуктов АМТ-ГРУП

В течение последних двух лет практически ни одно мероприятие или конференция по информационной безопасности не обходится без обсуждения вопросов разработки безопасного программного обеспечения. В профессиональном сообществе уже даже устоялся термин РБПО. Термин, во избежание множества толкований, был уточнён и зафиксирован в новом ГОСТ 56939–2024 как «содержание и порядок выполнения работ, связанных с созданием безопасного программного обеспечения и устранением выявленных недостатков, в том числе уязвимостей ПО».

Можно говорить, что РБПО — это процесс создания программного обеспечения, который включает в себя меры по обеспечению безопасности кода на всех этапах его жизненного цикла. Это актуально, несмотря на то что в вышеупомянутом ГОСТ 56939–2024 прямо отмечается, что *«поскольку модель жизненного цикла ПО зависит от специфики, масштаба, сложности ПО и условий, в которых ПО создаётся и функционирует, приведённые в ...стандарте процессы намеренно не связываются с конкретной моделью жизненного цикла ПО».*

Но, к сожалению, терминологические нюансы являются самой меньшей из проблем при внедрении РБПО.

Многие компании и отдельные команды разработки, даже те, которые напрямую не связаны с производством средств защиты, уже получили от своих заказчиков, внутреннего менеджмента и даже подразделений маркетинга требования к максимально оперативной организации РБПО. Причём часто срок реализации этих требований выходит за рамки разумного, вызывая только раздражение и негодование у исполнителей. С одной стороны, внедрение РБПО преподносится как очевидное и всеобщее благо. С другой стороны, процессы и процедуры его внедрения предлагаются настолько сжатыми по времени и ресурсам, что часто вызывают лишь удивление.

Попробуем разобраться, в чём же состоят сложности оперативного внедрения РБПО (далее «безопасная разработка») в жизнь обычного коллектива, команды или компании. Можно выделить несколько причин, почему компании не спешат внедрять безопасную разработку в свои процессы:

1. Сложность внедрения и отсутствие общедоступных полных и наглядных стандартов и рекомендаций. Внедрение безопасной разработки требует изменения существующих процессов разработки и эксплуатации, а также обучения сотрудников новым методам работы. Это может быть сложным и длительным процессом. Процесс требует квалифицированного менеджмента, инвестиций и должен опираться не просто на требования стандартов, а на примеры реализаций, референсные архитектуры и паттерны взаимодействия нескольких программных, а иногда и аппаратных решений. Не имея таких примеров, многие коллективы считают её чем-то слишком дорогим, организационно и технически сложным. Несмотря на появление очередного ГОСТ, конкретные практики внедрения безопасной разработки найти крайне сложно. Ещё сложнее собрать конвейер программных и технических средств, вычислительных мощностей, средств ИБ с учётом производства конкретного продукта. Отсутствие готовых решений затрудняет и замедляет процесс внедрения и вызывает справедливые опасения у тех команд, которые ещё только в начале пути.

2. Отсутствие понимания преимуществ и неопределённость результатов. Некоторые компании не понимают, какие конкретно преимущества на практике им даст внедрение безопасной разработки. Рассматривают культуру безопасной разработки просто как ещё один модный тренд, который не принесёт реальной пользы на практике. Особенно это актуально для компаний, чья технологическая экспертность сосредоточена в основном в отраслевых знаниях. Речь идёт о разработчиках АСУ

ТП систем, SCADA-систем, систем, применяемых в энергетике, продуктов в сфере консультационных услуг, программно-аппаратных решений «полевого уровня», IoT-решений. То есть таких решений, где надёжность функционирования и промышленная безопасность, точность получаемых данных ранее всегда превалировала над информационной безопасностью. Как и с процессами управления рисками, результаты внедрения безопасной разработки могут быть неопределёнными на старте, отложенными и часто неподтверждаемыми до тех пор, пока не наблюдаются конкретные инциденты безопасности в отношении разрабатываемого ПО. Часто компании опасаются, что они вообще не получат никаких результатов или что результаты будут достигнуты слишком медленно. Внедрение РБПО не воспринимается как что-то, что напрямую может увеличить прибыль, потому что в лучшем случае приводит к реализации нефункциональных требований продуктов, которые «продать рынку достаточно тяжело».

3. Сопротивление изменениям и страх перед ошибками. Любые изменения в процессах разработки и эксплуатации программного обеспечения могут вызвать сопротивление со стороны сотрудников. Специалисты могут опасаться, что внедрение новых методов работы приведёт к увеличению нагрузки или снижению их производительности. Более того, на первых этапах внедрения безопасной разработки именно так и происходит, что только подтверждает аргументы противников изменений. Несовершенство применяемых конвейеров сборки и систем тестирования, интеграционные проблемы на пути внедрения лишь добавляют в эту копилку свои аргументы против. Команды, менеджеры и отдельные специалисты могут бояться совершить ошибки при внедрении безопасной разработки. Ожидают, что новые процессы приведут к снижению безопасности их систем, навредят репутации, снизят прибыль. Эта проблема также актуальна в том случае, когда в уже существующие, хорошо отлаженные производственные конвейеры привносятся новые инструменты, решения, процессы.

4. Недостаток ресурсов. Внедрение безопасной разработки может потребовать дополнительных ресурсов, таких как время, деньги, квалифицированные специалисты, аппаратные мощности. Не все компании готовы выделить эти ресурсы. Чаще всего внедрение безопасной разработки воспринимается как ещё одна обязанность текущих команд. Делаются попытки автоматизировать дополнительно возникающие операции путём подбора «готового» программного обеспечения без какой-либо адаптации.

5. Необходимость пересмотра архитектуры системы. Часто архитектура разрабатываемых систем унаследована, закладывалась одним или несколькими ключевыми специалистами с учётом актуальных на момент проектирования задач (то есть «в прошлом»). Аспекты Security by Design могли откладываться на второй план или закрываться типовыми решениями: применением компонентов с открытым исходным кодом, базовыми фреймворками. В результате, когда для внедрения безопасной разработки необходимо пересмотреть архитектуру системы и внести в неё изменения, направленные на повышение уровня безопасности, это, с одной стороны, встречает сопротивление некоторых ключевых специалистов, а с другой — приводит к дополнительным затратам времени и денег. В ряде случаев внедрение РБПО вырождается в полный демонтаж старой архитектуры разрабатываемых систем.

6. Проблемы с интеграцией на организационном уровне и уровне корпоративной культуры. В основе идеи безопасной разработки лежит комплексная интеграция процессов разработки, эксплуатации и обеспечения безопасности в целом. Однако, как и в случае с унаследованной архитектурой систем, ряд новых процессов и систем не может быть внедрён в компании просто потому, что это может организационно, технологически и даже этически противоречить тем процессам, которые функционируют в организации сейчас. В разных компаниях могут существовать разные культурные нормы и ценности, правила, которые влияют на отношение к изменениям в целом и к вопросам защиты данных, применению средств защиты, разграничению доступа. В некоторых компаниях в принципе принят более консервативный подход к изменениям, что также замедляет внедрение безопасной разработки.

Однако вне зависимости от обозначенных ограничений, похоже, у внедрения процессов безопасной разработки в деятельность практически каждой организации-разработчика ПО нет альтернатив. С развитием технологий и увеличением количества киберугроз обеспечение безопасности разрабатываемого ПО становится всё более актуальной задачей. С внедрением процессов безопасной разработки компании смогут быстрее и эффективнее обнаруживать и устранять уязвимости в своих системах. Это позволит организациям снизить риски кибератак, защитить свои данные, системы и репутацию. Кроме того, уже очевидно, что безопасная разработка поможет компаниям соответствовать требованиям законодательства и стандартов безопасности, которые, как мы видим, ужесточаются с каждым днём.

* * *

Дискуссии по теме безопасной разработки, которые идут на рынке, уже сейчас способствуют развитию культуры безопасности в компаниях. Благодаря таким дискуссиям разработчики, менеджеры, заказчики, операторы систем и эксплуатирующие организации, да и сами специалисты ИБ лучше понимают важность обеспечения безопасности в своих продуктах.

ЗАВИСИТ ОТ ЦЕЛИ

ОЦЕНКА ПРОЦЕССОВ РАЗРАБОТКИ БЕЗОПАСНОГО ПО

Антон СВИНЦИЦКИЙ
директор по консалтингу
АО «ДиалогНаука»

Сергей КАНИВЕЦ
ведущий консультант отдела
консалтинга АО «ДиалогНаука»

Безопасность программного обеспечения (ПО) играет критически важную роль в условиях современной цифровой экономики. Регулярные кибератаки, утечки данных и нарушения конфиденциальности негативно влияют как на коммерческие организации, так и на государственные структуры. Одним из ключевых факторов повышения уровня защиты является корректная организация процессов разработки безопасного ПО. Однако разработка безопасного ПО — это не только внедрение технических мер защиты, но и организация процесса с акцентом на управление рисками, соответствие требованиям регуляторов и обучение специалистов. В данной статье рассмотрены основные методики и подходы к оценке процессов безопасной разработки.

МЕТОДИКИ ОЦЕНКИ ПРОЦЕССОВ РАЗРАБОТКИ БЕЗОПАСНОГО ПО

Оценка процессов разработки безопасного ПО является важным этапом для создания защищённых программных продуктов, направленным на проверку соответствия процессов современным стандартам, выявление слабых мест и разработку корректирующих мер. Сейчас уже существует множество подходов и инструментов для оценки зрелости процессов разработки, например OWASP SAMM, DevSecOps Maturity Model и Synopsys BSIMM, позволяющие не только оценить текущее состояние, но и наметить пути его улучшения. Рассмотрим особенности каждой из этих моделей.

OWASP Software Assurance Maturity Model (SAMM) — это нормативная или предписывающая модель, которая структурирует процессы разработки безопасного ПО. Она включает 15 практик, разделённых на пять бизнес-функций: Governance, Design, Implementation, Verification и Operation. Каждая практика состоит из двух потоков: A — «поверхностный» и B — «углублённый», что позволяет выбирать степень детализации. SAMM обеспечивает формализацию и измерение результатов, минимизируя затраты на оценку. Компания самостоятельно определяет целевой уровень зрелости в удобном для неё горизонте планирования: от одного до пяти лет. Для автоматизации оценки компания может использовать доступные инструменты, такие как SAMMwise или SAMMY.

OWASP DevSecOps Maturity Model (DSOMM) описывает практики, направленные на интеграцию безопасности в процессы разработки и CI/CD. Модель выделяет пять инструментов: Build and Deployment, Culture and Organization, Implementation, Information Gathering, Test and Verification, каждый из которых включает меры защиты, распределённые по пяти уровням зрелости.

Synopsys BSIMM — это описательная модель, базирующаяся на практике 130 организаций. Она состоит из четырёх доменов, каждый из которых включает три практики, разделённые на три уровня покрытия. BSIMM позволяет компаниям сравнивать свои достижения с общепринятыми в индустрии практиками, реализованными у лидеров рынка, а доступные инструменты облегчают процесс оценки.

КОНЦЕПЦИЯ ОЦЕНКИ ПРОЦЕССА РАЗРАБОТКИ БЕЗОПАСНОГО ПО

Центральное место в этой концепции занимает системный подход, который позволяет структурировать и интегрировать меры безопасности на всех уровнях. В основе такого подхода лежит идея цикличности и непрерывного совершенствования, что обеспечивает устойчивость к новым угрозам и адаптацию к меняющимся условиям.

Управление процессом безопасной разработки является основой для обеспечения системного подхода к защите программного обеспечения. Оно охватывает ключевые аспекты управления — от стратегического планирования до контроля и оценки соответствия, что обеспечивает комплексное внедрение мер безопасности на всех этапах разработки, в том числе позволяет оценить текущее состояние процессов по следующим направлениям:

- описание стратегии и планирование процесса разработки (так как эффективное управление начинается с формулирования стратегии, которая определяет долгосрочные цели

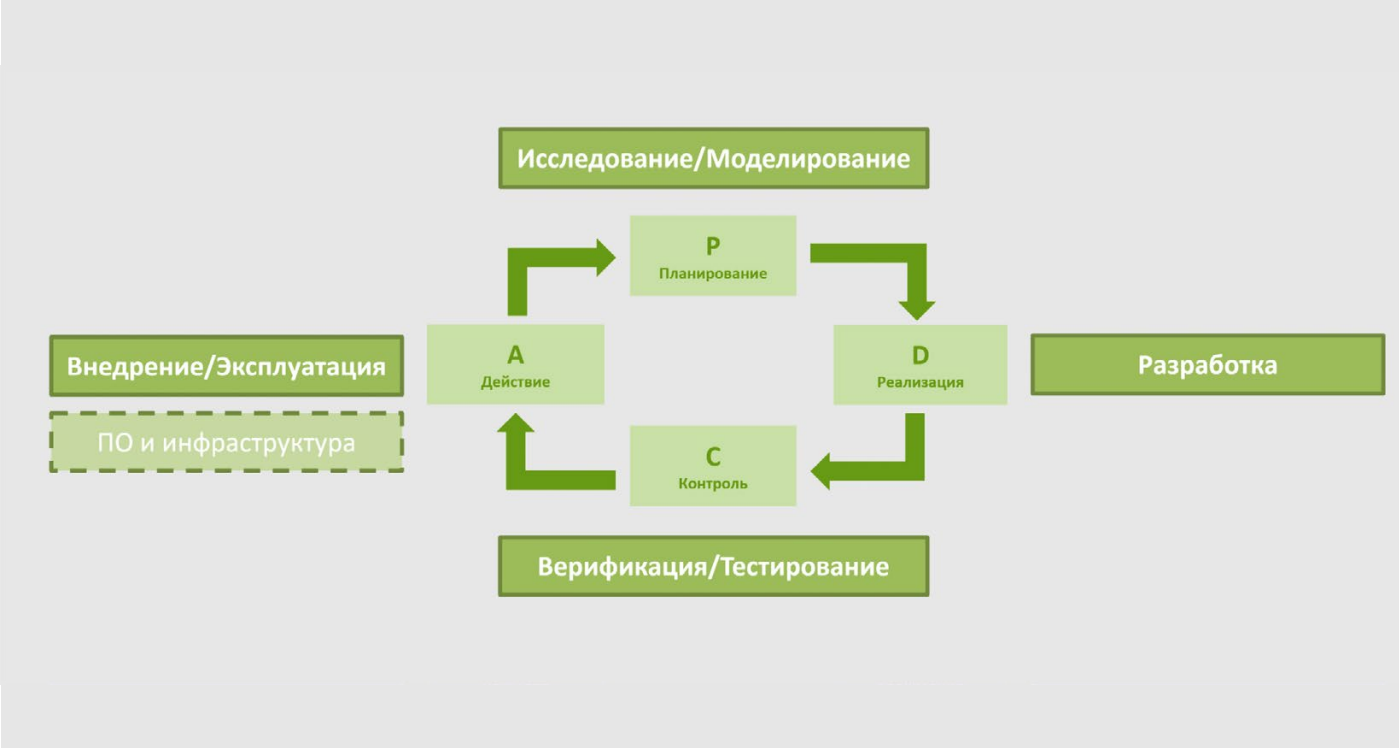


Рисунок 1. Соответствие элементов цикла PDCA мерам, относящимся к процессам безопасной разработки ПО

и задачи, в том числе в области разработки безопасного ПО);

- метрики эффективности (позволяют измерить и оценить текущий уровень безопасности разработки, а также отслеживать прогресс);
- повышение осведомлённости и квалификации (обучение персонала играет ключевую роль в предотвращении ошибок, связанных с безопасностью);
- контроль (контроль должен охватывать все аспекты процессов разработки и помогает обеспечить соблюдение установленных стандартов);
- оценка соответствия (эта часть включает проверку соответствия процессов безопасной разработки установленным стандартам и законодательным требованиям).

Цикл PDCA (планирование — реализация — контроль — действие) является универсальной моделью управления процессами, которая может быть применима и к процессам разработки безопасного ПО. В контексте обеспечения безопасности PDCA помогает систематизировать и интегрировать защитные меры на всех этапах жизненного цикла продукта (рис. 1).

Каждый элемент цикла включает набор ключевых мер и действий, по которым оцениваются процессы:

1. Планирование (PLAN) — этап исследования и моделирования:

- подготовка требований;
- моделирование угроз / оценка рисков;
- требования безопасности при использовании зависимостей;
- проектирование архитектуры.

2. Реализация (DO) — этап разработки ПО:

- управление конфигурацией ПО;
- требования к безопасности системы сборки и раскатки ПО;
- безопасность инфраструктуры разработки и тестирования;
- обеспечение целостности кода.

3. Контроль (CHECK) — этап верификации и тестирования:

- оценка архитектуры;
- анализ кода;
- тестирование защищённости.

4. Действие (ACT) — этап внедрения и эксплуатации:

- внедрение и эксплуатация (подготовка, доставка, вывод из эксплуатации);
- безопасность инфраструктуры (эксплуатация инфраструктуры ПО, тестирование на проникновение, управление уязвимостями, управление инцидентами).

ВЫВОДЫ

Выбор методики оценки зависит от целей организации: стремление достичь определённого уровня зрелости или соответствие международным стандартам. Использование моделей OWASP SAMM, DSOMM или BSIMM позволяет выявить слабые места в процессах и разработать долгосрочные стратегии совершенствования. Это обеспечивает соответствие требованиям безопасности, сокращает риски и укрепляет доверие пользователей. Представленная концепция возможна к применению и для новой редакции национального стандарта ГОСТ Р 56939–2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования».

Применение таких подходов делает процессы разработки ПО более надёжными и способствует достижению высокого уровня защищённости, соответствующего мировым стандартам.

ОБНОВЛЕНИЕ ОС «АЛЬТ СП» РЕЛИЗ 10

СЕРТИФИЦИРОВАННАЯ СУБД, ПОДДЕРЖКА СОВРЕМЕННЫХ ПРОЦЕССОРОВ «ЭЛЬБРУС»

Константин ГЕРАЩЕНКО
редактор «Базальт СПО»

«Базальт СПО» выпустила сертифицированное ФСТЭК России обновление операционной системы «Альт СП» релиз 10. В новую версию вошла сертифицированная СУБД PostgreSQL дополнительно к сертифицированным средствам виртуализации и контейнеризации. Появились сборки для процессоров «Эльбрус 2С3» и «Эльбрус 8СВ».

Согласно новым требованиям ФСТЭК России в состав операционной системы включены программные интерпретаторы (php, perl, lua, python, nodejs) и веб-сервер (nginx), прошедшие испытания по выявлению уязвимостей и недекларированных возможностей в ПО в соответствии с методикой ФСТЭК России.

Сертифицированная СУБД в составе ОС. В «Альт СП» включена система управления базами данных **PostgreSQL версии 16.6,** сертифицированная на соответствие приказу № 64 ФСТЭК России по 4 классу защиты. При этом защищенными являются не только одиночные установки СУБД, но и кластеры из нескольких баз данных.

Архитектуры, поддерживаемые «Альт СП». Сборки для архитектур x86_64 и aarch64 традиционно выпущены в исполнении «Рабочая станция» и «Сервер». В сборку для aarch64 добавлено ядро с поддержкой устройств на Rockchip 3588.

В обновленной версии появились сборки для рабочих станций для процессоров «Эльбрус 8СВ» и «Эльбрус 2С 3». Сборки для процессоров «Эльбрус 4С» и «Эльбрус 8С», созданные на основе 9-й платформы, входят в выпуск с ограниченной поддержкой (до августа 2026 года). Для тех, кто хочет перейти на релиз 10 для процессора «Эльбрус 8С», можно использовать образ для процессора «Эльбрус 8СВ».

Обновление установщика системы. В новой версии инсталлятор позволяет выбирать нужные приложения на этапе установки. Это упрощает первоначальную настройку системы, особенно для пользователей, которым нужна минимальная конфигурация для использования средств виртуализации или контейнеризации (рис. 1).

Добавлен режим OEM-установки, благодаря чему поставщики оборудования могут массово предустанавливать ОС на свои устройства.

Обновление средств управления контейнерами. По-прежнему используются инструменты **podman** (обновлен до 4.9.4) и **kubernetes** (обновлен до 1.31), доработан набор скриптов **podsec**: действия, которые раньше выполнялись вручную, теперь автоматизированы. В репозитории доступен набор пакетов kubernetes от версии 1.22 до 1.31, что позволяет пользователям плавно перейти к новому ПО.

Обновлены и входящие в состав «Альт СП» контейнеры. К набору контейнеров для разворачивания кластера kubernetes добавлены базовые контейнеры, в том числе и distroless-образы. Получить образы и их обновления также можно через сервис registry.altlinux.org (по тегу c10f2).

Обновление средств управления виртуальными машинами. Обновлена система виртуализации на основе Proxmox Virtual Environment (PVE). До 17 версии обновлена файловая система ceph — программная объектная сеть хранения, обеспечивающая как файловый, так и блочный интерфейсы доступа.

Добавлен модуль Linstor для управления блочными устройствами хранения данных. Linstor размещает тома для хранения данных и включает для них репликацию. Добавлена библиотека для программно-определяемой сети.

Для повышения отказоустойчивости кластера PVE добавлена служба Corosync Quroum, позволяющая кластеру выдержать большее количество отказов узлов, чем это позволяют стандартные правила кворума.

Программный комплекс «Альт Домен» в составе ОС. В состав исполнения Сервер входит программный комплекс для централизованного управления компьютерами и учетными записями пользователей «Альт Домен» (аналог Microsoft Active Directory), его можно выбрать на этапе установки как отдельную группу пакетов. В исполнении Рабочая станция есть группы «Клиент домена» и «Инструменты управления групповыми политиками», с их помощью развертывание программного комплекса «Альт Домена» упрощается.

Появилась система идентификации и управления доступом **Keycloak**. Она используется для аутентификации и авторизации как через локальные серверы, так и через соцсети и другие ресурсы, а также поддерживает двухфакторную аутентификацию и технологию единого входа SSO (Single Sign-On). Позволяет войти в несколько связанных сервисов с единым логином и паролем благодаря привязке к корпоративным системам хранения учётных записей.

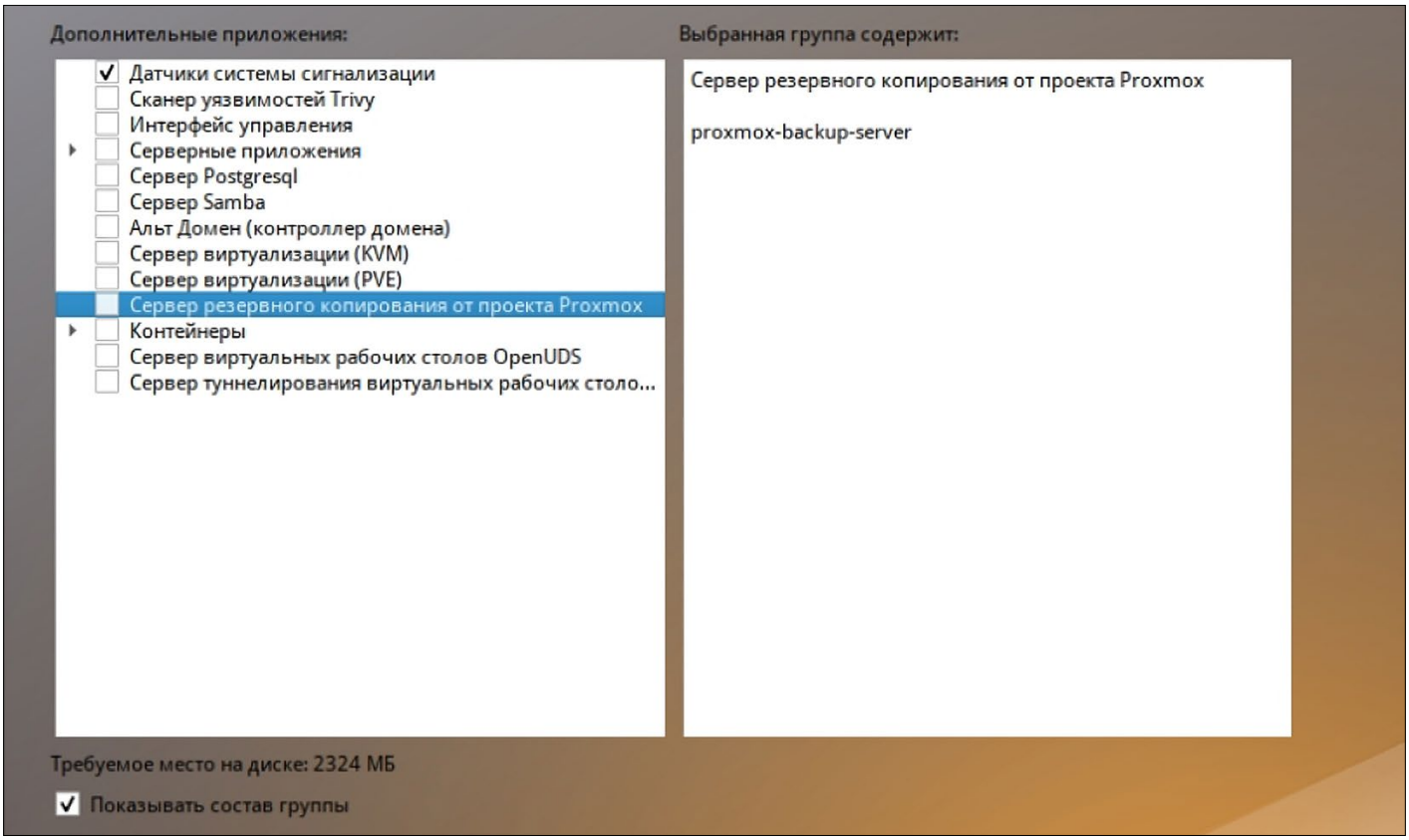


Рисунок 1. В новой версии инсталлятор позволяет выбирать нужные приложения на этапе установки

Новая система мониторинга событий безопасности. В состав инструментов мониторинга событий безопасности добавлено современное гибкое средство **icinga2**, которое можно использовать вместо **nagios**.

Безопасное использование USB-устройств. Основные функции по контролю и ограничению действий пользователя с USB-устройствами выполняют пакеты Alterator-usbguard и Alterator-usbmount, которые доступны через Вебверсию Центра управления системой.

Можно задавать правила для разрешения или блокировки устройств по их характеристикам, например, по идентификаторам или по интерфейсам. Администратор может управлять политиками через удобный интерфейс, импортировать настройки и аудит событий.

Добавлена возможность ограничивать доступ к файловой системе USB-устройств, можно настраивать права доступа для пользователей и групп и файловые системы для контроля доступа.

Добавлен новый механизм работы в режиме киоска. Он реализован в виде модуля Alterator, работает в графической среде рабочего стола и позволяет ограничивать возможности запуска программ по профилям, подготовленным администратором системы. В систему включены примеры таких профилей, которые можно использовать в качестве образцов.

Интерфейс пользователя. В «Альт СП» по умолчанию устанавливается тема «как windows», что делает внешний вид системы привычным пользователю.

Новый набор иконок поможет администратору сразу определить, обновлена ли система.

Преднастроена виртуальная клавиатура для окна входа и разблокировки хранителя экрана.

Появилось приложение, позволяющее администратору удаленно подключаться к рабочему столу пользователя при использовании графической оболочки MATE.

Новые функциональные возможности, обновление Java. Добавлено расширение к системному файловому менеджеру, которое позволяет высчитывать и сравнивать контрольные суммы файлов, в том числе и по алгоритму ГОСТ Р 34.11–2012.

В рабочую станцию включен **recoll** — инструмент для поиска информации в документах, архивах, и письмах по различным параметрам, в том числе по фрагменту текста.

Добавлена LTS-версия **Java 21**.

Совместимость. В ОС включен модуль **cryptopro-preinstall** (только для архитектуры x86_64), упрощающий установку официальных пакетов КриптоПро CSP (с поддержкой Rutoken S и ECP). Добавлены корневые сертификаты для сайтов от российского центра сертификации ООО «ТЦИ».

Обеспечена поддержка «Сигнатуры-L» — системы криптографической авторизации электронных документов, разработанной Банком России. Она предназначена для криптографической защиты электронных сообщений в среде операционных систем Linux.

Также реализована полноценная поддержка Медицинской Информационной Системы «САМСОН».