Безопасная разработка

ТЕХНИЧЕСКОЕ ОПИСАНИЕ БЕЗОПАСНАЯ РАЗРАБОТКА ФСТЭК / ГОСТ Р 56939

Безопасная разработка НАЙС.ОС по требованиям ФСТЭК

Документ описывает организацию безопасной разработки НАЙС.ОС как совокупность процессов, контролей и артефактов жизненного цикла ПО: от формирования требований и анализа угроз до испытаний безопасности, выпуска подписанных обновлений и обработки уязвимостей. Описание ориентировано на проверяемость (аудитопригодность) и трассируемость «требование реализация проверка релиз».

Назначение документа

- Зафиксировать требования и правила выполнения работ по безопасной разработке НАЙС.ОС в терминах ФСТЭК и ГОСТ, включая контрольные мероприятия и ожидаемые результаты.
- Описать набор артефактов, подтверждающих выполнение процесса: политики, модели угроз, архитектурные описания, отчёты испытаний, релизные чек-листы, процедуры управления уязвимостями, порядок доставки обновлений.
- Обеспечить единый формат представления данных для заказчиков, аудиторов и инженеров сопровождения.

Целевая аудитория

- Заказчики и подразделения ИБ (в т.ч. для контуров КИИ): оценка достаточности процесса, состава подтверждающих материалов и порядка обновления.
- Аудиторы и партнёры: проверка соответствия заявленного процесса фактическому выполнению на основании артефактов и записей.
- Команды разработки/сборки/релиз-инжиниринга НАЙС.ОС: единые правила

внесения изменений, критерии готовности, обязательные проверки и ответственность.

Результаты для читателя

- Понимание структуры SSDLC/БРПО для НАЙС.ОС и состава обязательных активностей.
- Перечень артефактов и доказательств выполнения процесса, пригодных для предоставления на проверку.
- Точки контроля качества и безопасности: анализ угроз, архитектура, SAST/DAST, фаззинг, релизные проверки.
- Порядок поддержки безопасности: обработка уязвимостей, уведомления, выпуск и проверка обновлений.

ВАЖНО
Границы и допущения
Документ не является заявлением об отсутствии уязвимостей. Цель — описание управляемого процесса, снижающего вероятность дефектов и обеспечивающего регламентированное обнаружение, исправление, выпуск и доставку обновлений с контролем целостности и подлинности.

1. Нормативная рамка: на что мы опираемся

ФСТЭК / ГОСТ

Раздел определяет нормативные источники, используемые для построения и проверки процесса безопасной разработки НАЙС.ОС (SSDLC/БРПО) и процесса поддержки безопасности (управление уязвимостями, выпуск и доставка обновлений). Нормативная рамка применяется для формирования проверяемых требований к работам и артефактам на всех стадиях жизненного цикла.

Ограничение применения
Раздел предназначен для инженерного и аудиторского использования (определение работ, артефактов, критериев и проверок). Текст не является юридическим толкованием нормативных актов.
Принцип применения требований
Требования рассматриваются как проверяемые. Для каждого требования определяется: (1) набор работ, (2) ожидаемый результат, (3) подтверждающий артефакт (документ, отчёт, протокол, лог проверки, релизный чек-лист).

Трассируемость фиксируется по схеме `требование → реализация → проверка → релиз`.

1.0 Перечень применяемых документов

Документ	Назначение в процессе НАЙС.ОС	Ожидаемые результаты (инженерная интерпретация)
ГОСТ Р 56939-2024 Национальный стандарт	Базовый стандарт для построения процесса безопасной разработки и устранения недостатков/уязвимостей. Используется для определения состава работ, этапов, ролей и артефактов.	Формализованный SSDLC/БРПО; фиксируемые результаты работ; управляемая поддержка безопасности после выпуска. Ссылка: карточка ГОСТ
Приказ ФСТЭК №239 Требования для значимых объектов КИИ	Источник обязательных требований к ПО, применяемому в КИИ: требования к БРПО, испытаниям по выявлению уязвимостей и поддержке безопасности.	Наличие руководства по БРПО и анализа угроз; выполнение испытаний (SAST, fuzzing, при необходимости DAST); наличие процедур исправления и уведомления; доставка обновлений с проверкой целостности/подлинности. Ссылка: официальная публикация
Приказ ФСТЭК №240 Сертификация процессов БРПО (ПО СЗИ)	Регламент сертификации процессов проектирования и производства ПО СЗИ. Используется как ориентир к полноте материалов и аудитопригодности процесса.	Формирование области процесса, состава материалов, записей и результатов контроля, достаточных для подтверждения соответствия. Ссылка: официальная публикация

Документ	Назначение в процессе НАЙС.ОС	Ожидаемые результаты (инженерная интерпретация)
Регламент БДУ (уязвимости) Порядок включения/обработки сведений	Регламент обмена сведениями об уязвимостях и требований к составу данных. Используется для построения VDP/PSIRT: приём, triage, выпуск исправлений, раскрытие, фиксация статусов.	Стандартизация состава сведений по уязвимости и дисциплина обработки (регистрация, подтверждение, оценка, статус, публикация). Ссылка: регламент на сайте БДУ

1.1 ГОСТ Р 56939-2024

ГОСТ Р 56939-2024 используется как базовый стандарт для описания и нормирования работ по безопасной разработке ПО и устранению выявленных недостатков. В контуре НАЙС.ОС стандарт применяется для формирования внутренней структуры процесса: этапы, входные данные, выходные артефакты, роли и контрольные мероприятия.

- **Нормирование процесса:** определение набора обязательных активностей SSDLC/БРПО на стадиях требований, проектирования, реализации, верификации, релиза и поддержки.
- **Артефакты и записи:** формирование документов и результатов проверок, достаточных для аудита (политики, модели угроз, архитектурные описания, отчёты испытаний, релизные протоколы).
- **Критерии готовности:** формализация условий, при которых выпуск допускается или блокируется (дефекты, результаты тестов, несоответствие требованиям).
- **Поддержка безопасности:** обеспечение управляемой обработки уязвимостей и выпуск исправлений после релиза.

Источник: карточка ГОСТ Р 56939-2024

1.2 Приказ ФСТЭК №239 (КИИ)

Приказ ФСТЭК №239 применяется как нормативная основа для требований к программному обеспечению, используемому в значимых объектах КИИ. В инженерной постановке документ фиксирует ожидаемые элементы процесса:

документирование БРПО, испытания по выявлению уязвимостей и требования к поддержке безопасности. Эти требования используются для определения минимального набора обязательных артефактов и проверок.

- **Безопасная разработка:** наличие руководства по БРПО; выполнение анализа угроз безопасности информации ПО; при необходимости — архитектурное описание на уровне подсистем и сопоставление функций/интерфейсов с подсистемами.
- **Испытания по выявлению уязвимостей:** выполнение статического анализа исходного кода (SAST) и фаззинг-тестирования; при необходимости — динамический анализ (DAST).
- **Поддержка безопасности:** наличие процедур исправления ошибок/уязвимостей; регламент уведомлений; порядок получения обновлений; контроль целостности и подлинности обновлений.

Источник: официальная публикация приказа №239

1.3 Приказ ФСТЭК №240 (сертификация процессов БРПО для ПО СЗИ)

Приказ ФСТЭК №240 устанавливает порядок сертификации процессов безопасной разработки ПО средств защиты информации. Для НАЙС.ОС документ используется как ориентир к представлению процесса в форме, пригодной для внешней оценки: определение области процесса, состава материалов и достаточности доказательств выполнения требований.

- **Аудитопригодность:** наличие формализованных описаний процессов, ролей, контрольных точек и правил принятия решений.
- **Комплектность материалов:** наличие документов и записей, позволяющих подтвердить выполнение требований на релизе и на выборке изменений.
- **Управляемость процесса:** фиксация результатов контроля, регистрация несоответствий и корректирующих мероприятий.

Источник: официальная публикация приказа №240

1.4 Регламент БДУ: обмен сведениями об

уязвимостях

Регламент БДУ применяется для стандартизации взаимодействия по сведениям об уязвимостях и построения процесса управления уязвимостями как части поддержки безопасности. В рамках НАЙС.ОС регламент используется для определения минимально необходимого состава данных, фиксируемых при обработке уязвимости, и требований к дисциплине учёта статусов (регистрация, подтверждение, оценка, исправление, публикация).

- **Приём и регистрация:** единый канал приёма сообщений, регистрация обращения и идентификатор инцидента/задачи.
- **Triage:** подтверждение воспроизводимости, определение затронутых версий/условий, предварительная оценка критичности.
- **Исправление и проверка:** подготовка патча, регрессионная проверка, выпуск обновления.
- **Раскрытие и уведомление:** публикация advisory (затронутые версии, меры снижения риска, порядок обновления), обеспечение проверяемой поставки обновления.

Источник: регламент на сайте БДУ ФСТЭК

Результат применения нормативной рамки
Нормативные источники определяют минимальный состав управляемых практик: (1) формализованная безопасная разработка, (2) испытания по выявлению уязвимостей (SAST/fuzz/при необходимости DAST), (3) поддержка безопасности (управление уязвимостями, уведомления, обновления с проверяемой целостностью/подлинностью), (4) аудитопригодность процесса и комплектность подтверждающих материалов.

2. Термины (определения для инженерного применения)

ГЛОССАРИЙ

Термины в этом разделе используются в дальнейшем тексте строго в приведённом смысле. Определения ориентированы на практическое применение при разработке, сборке, испытаниях, релизе и сопровождении компонентов НАЙС.ОС, а также при подготовке материалов для аудита.

Замечание о трактовке
В контексте требований ФСТЭК безопасность разработки рассматривается как **управляемый процесс** и **проверяемые артефакты** (правила, отчёты, протоколы, результаты испытаний). Инструменты (анализаторы, сканеры, фаззеры) являются средствами реализации процесса и не заменяют его.

2.1 Безопасная разработка / SSDLC / БРПО

`Безопасная разработка ПО`
: Совокупность регламентированных работ в жизненном цикле ПО, направленных на предотвращение, выявление и устранение недостатков безопасности до выпуска и после выпуска. Ключевой признак — наличие фиксируемых результатов и записей, позволяющих проверить факт выполнения работ и их полноту.

`SSDLC` (Secure Software Development Lifecycle)
: Жизненный цикл разработки с встроенными контрольными мероприятиями безопасности. Практически означает, что безопасность присутствует на этапах: требования проектирование реализация верификация релиз поддержка безопасности. На каждом этапе определены обязательные проверки и критерии допуска к следующему этапу.

`БРПО` (безопасная разработка программного обеспечения)
: Термин, используемый в регуляторном контексте для обозначения формализованного процесса безопасной разработки. В рамках статьи БРПО трактуется как SSDLC, дополненный требованиями к доказательности: **политики**, **модель угроз**, **архитектурные описания**, **результаты испытаний**, **релизные протоколы** и **процедуры поддержки безопасности**.

Практический критерий “процесс vs инструмент”
Наличие статического анализатора или сканера уязвимостей не является доказательством БРПО. Доказательством является: (1) установленная обязательность выполнения проверок, (2) определённые пороги/критерии (что блокирует релиз), (3) хранение результатов и решений, (4) воспроизводимость проверки на релизной ветке.

2.2 SAST / DAST / fuzzing

`SAST` (Static Application Security Testing)

Статический анализ исходного кода или промежуточного представления без выполнения программы. Цель — выявление классов ошибок безопасности по структуре кода и потокам данных.
Типовые классы дефектов, которые выявляет SAST:

- использование небезопасных API и анти-паттерны обработки входных данных;
- потенциальные переполнения/выходы за границы, use-after-free (для C/C++), ошибки управления памятью;
- инъекции (SQL/command) и недостаточная валидация/экранирование (в зависимости от языка и правил);
- ошибки криптоприменения (слабые режимы/параметры) — при наличии соответствующих правил анализа;
- утечки секретов (при подключении правил поиска секретов) и небезопасные практики логирования.

`DAST` (Dynamic Application Security Testing)
Динамическое тестирование безопасности на выполняющемся экземпляре (служба/приложение/компонент), как правило через внешние интерфейсы (HTTP API, CLI, сетевые протоколы). Цель — обнаружение уязвимостей, проявляющихся только при выполнении и взаимодействии компонентов.
Типовые классы дефектов, которые выявляет DAST:

- ошибки аутентификации и авторизации на реальных маршрутах/эндпоинтах;
- инъекции и небезопасная обработка входных данных на уровне работающих обработчиков;
- ошибки конфигурации безопасности (заголовки, TLS-параметры, доступы, режимы отладки);
- уязвимости, связанные с интеграциями и зависимостями на рантайме (прокси, БД, очереди);
- проблемы управления сессиями и утечки данных через ответы/ошибки.

`Fuzzing` (фаззинг-тестирование)
Метод тестирования, при котором компонент получает большое количество автоматически генерируемых или мутированных входных данных (файлы, пакеты, API-запросы, протокольные сообщения) для выявления отказов и некорректного поведения. Фокус — на устойчивости к некорректному вводу и поиске дефектов, приводящих к авариям или нарушению памяти.
Что фаззинг находит наиболее эффективно:

- краши, зависания, некорректные исключения и неконтролируемое потребление ресурсов;
- ошибки парсинга форматов/протоколов и “неучтённые состояния” конечных автоматов;
- дефекты управления памятью (для C/C++), которые могут быть эксплуатируемыми;
- ошибки обработки границ (длины, смещения, кодировки, вложенность структур).

2.3 Модель угроз и поверхность атаки

Модель угроз
: Формализованное описание активов (что защищаем), границ доверия (где меняются уровни доверия), потенциальных нарушителей (кто атакует), векторов атак (как атакуют) и мер снижения риска. Модель угроз используется для обоснования архитектурных решений, выбора обязательных проверок и определения приоритета дефектов.

Поверхность атаки
: Совокупность всех входов и наблюдаемых воздействий на компонент: сетевые порты и протоколы, API, форматы файлов, IPC, привилегированные операции, точки расширения (плагины), пути конфигурации и обновления. Поверхность атаки определяет, какие проверки обязаны быть выполнены и где требуется усиление изоляции/минимизация прав.

Почему проверки “слепые” без модели угроз
Без модели угроз невозможно определить полноту испытаний: какие интерфейсы должны быть покрыты тестами, какие сценарии являются критическими, какие данные считаются недоверенными, и какие последствия имеют дефекты. В результате проверки превращаются в набор несвязанных запусков инструментов без критериев достаточности.

2.4 Поддержка безопасности

Поддержка безопасности (security maintenance)
: Регламентированный процесс обработки ошибок и уязвимостей после выпуска: приём сообщений, triage, исправление, проверка, выпуск обновления, уведомление пользователей и контроль поставки.

Уязвимость

Дефект, который при определённых условиях может приводить к нарушению конфиденциальности, целостности или доступности. Для инженерного процесса важны: воспроизводимость, затронутые версии/конфигурации, условия эксплуатации и оценка критичности.

Advisory (уведомление о безопасности)

Публикуемое уведомление, описывающее уязвимость и действия пользователя. Минимальный состав: идентификатор, описание, затронутые версии, влияние, меры снижения риска (mitigation), наличие исправления, инструкция обновления, контроль целостности/подлинности обновления.

Патч / исправление

Изменение кода/конфигурации/пакетирования, устраняющее дефект безопасности. Для процесса важны: трассируемость (связь с задачей/уязвимостью), рецензирование, тестирование и перенос исправления в поддерживаемые ветки.

Каналы обновлений

Определённые каналы поставки обновлений (например, stable/LTS/testing), различающиеся политикой попадания изменений, скоростью доставки и критериями допуска. Каналы являются частью управления риском и предсказуемости эксплуатации.

Проверка целостности и подлинности обновлений

Механизмы, подтверждающие, что обновление не было изменено по пути поставки (целостность) и что оно выпущено доверенной стороной (подлинность). Реализуется через криптографическую подпись, проверку хэшей и контроль доверенных ключей/репозиториев в процессе обновления.

3. Область действия: что считается “продуктом” НАЙС.ОС

SCOPE

Раздел устанавливает границы продукта НАЙС.ОС для целей безопасной разработки, испытаний, выпуска и поддержки безопасности. Определение области действия используется для исключения неоднозначностей при аудите: какие компоненты и артефакты относятся к продукту, какие процессы подтверждаются, и какие элементы находятся вне периметра ответственности производителя.

Определение “продукта” в контексте настоящего документа

Под “продуктом НАЙС.ОС” понимается совокупность: (1) исходных и сборочных спецификаций, (2) собранных пакетов и репозиториев, (3) поставляемых образов/коммитов/контейнерных слоёв, (4) инфраструктуры выпуска (подпись, публикация), (5) сопроводительных материалов релиза (метаданные, хэши/подписи, документация), находящихся под управлением производителя НАЙС.ОС и выпускаемых как единый релиз/версия.

3.1 Состав продукта НАЙС.ОС

В область действия включаются следующие классы компонентов. Для каждого класса определяются: правила изменения, обязательные проверки, результаты испытаний и артефакты выпуска.

Класс компонента	Состав (что включается)	Результат/артефакты выпуска
Базовая ОС	Ядро, init/система инициализации, системные сервисы и утилиты, сетевой стек, базовые библиотеки, криптографические библиотеки и провайдеры (при наличии), политики безопасности и дефолтные настройки, влияющие на модель угроз и поверхность атаки.	Пакеты базовой системы, конфигурационные профили по умолчанию, релизные метаданные и журналы изменений.
Репозитории пакетов (RPM)	Исходные спецификации сборки (spec), патчи, правила упаковки и именования, зависимости, флаги hardening, скрипты установки/удаления, системные пользователи/права (если применимо), метаданные репозитория.	Подписанные RPM-пакеты, репозиторные метаданные, сведения о версиях и сборках, отчёты контроля сборки.
Поставляемые образы и слои	Установочные и эксплуатационные артефакты: ISO, RAW (дисковые образы), OSTree-коммиты/деплои (если применимо), контейнерные базовые образы (base images) и производные минимальные слои.	Подписанные и/или снабжённые контрольными суммами артефакты поставки, манифесты, метаданные релиза.

Класс компонента	Состав (что включается)	Результат/артефакты выпуска
Инструменты сборки и CI	Конфигурация сборочной среды, пайплайны CI/CD, правила “чистой” сборки, контроль исходников и зависимостей, тестовые стенды для испытаний, наборы задач (lint/тесты/анализ/фаззинг), процедуры релизного утверждения.	Логи сборки и тестов, отчёты SAST/DAST/fuzzing (где применимо), протоколы релизных проверок.
Подпись и публикация	Инфраструктура криптографической подписи пакетов/репозиториев/образов, управление ключами, процедура публикации релизов и обновлений, каналы обновлений (stable/LTS/testing — при наличии), контроль целостности и подлинности поставки.	Подписи, ключевые материалы для проверки (публичные ключи), инструкции по проверке, опубликованные каналы обновлений и правила их использования.

3.2 Границы периметра ответственности

Периметр ответственности производителя НАЙС.ОС распространяется на компоненты и артефакты, перечисленные в п. 3.1, при условии их получения потребителем из официальных каналов поставки и без модификаций, нарушающих воспроизводимость и проверяемость артефактов.

Условия выхода за периметр
Следующие действия переводят экземпляр системы за пределы периметра, описанного в настоящем документе, если не предусмотрено иное отдельным соглашением/процедурой:

- локальная пересборка пакетов без использования утверждённой сборочной инфраструктуры и без сохранения артефактов проверки;
- внесение неучтённых патчей/изменений в спецификации, конфигурации безопасности или состав репозиториев;
- использование сторонних репозиториев и бинарных пакетов, не прошедших контроль цепочки поставки НАЙС.ОС;
- модификация образов/OSTree-деплоев/контейнерных базовых слоёв без повторной подписи и без фиксации метаданных релиза.

3.3 Элементы вне области действия

Следующие элементы не считаются частью продукта НАЙС.ОС в рамках настоящего документа и не включаются в объём подтверждения процесса безопасной разработки, за исключением явно оговорённых случаев (например, когда компонент включён в официальный состав поставки):

- **Сторонние приложения “upstream”** как проекты: их разработка и внутренние процессы не контролируются производителем НАЙС.ОС.
- **Пользовательские нагрузки**: прикладные сервисы заказчика, контейнеры, VM, конфигурации приложений и их секреты.
- **Интеграции и окружение эксплуатации**: внешние прокси, балансировщики, сторонние IdP, внешние БД и сервисы, если они не поставляются НАЙС.ОС.
- **Локальные модификации** системы заказчиком, не оформленные как управляемая ветка/релиз с артефактами проверки.
- **Средства разработки заказчика**: IDE, локальные сборочные хосты и сторонние CI, не входящие в утверждённую инфраструктуру НАЙС.ОС.

Практический эффект определения области действия
Чёткая фиксация состава продукта и границ ответственности используется для: (1) формирования полного перечня артефактов релиза и обновлений, (2) определения обязательных проверок и испытаний по каждому классу компонентов, (3) подготовки материалов для аудита и проверки цепочки поставки, (4) управления изменениями: исключения, расширения периметра и условия поддержки фиксируются явно.

4. Принципы НАЙС.ОС: системный подход к безопасности

PRINCIPLES

Раздел фиксирует принципы, на основании которых в НАЙС.ОС принимаются инженерные решения в области безопасности. Принципы применяются ко всем классам компонентов, входящим в область действия (см. раздел 3), и используются как критерии: (1) выбора архитектурных решений, (2) определения обязательных проверок, (3) оценки допустимости изменений при релизе и сопровождении.

4.1 Secure-by-design

Безопасность рассматривается как свойство архитектуры и требований. Для каждого компонента, включаемого в продукт, определяются требования безопасности, границы доверия, поверхность атаки и типовые сценарии нарушителя. Контрольные мероприятия (проверки, испытания, ограничения конфигурации) выбираются на основании этих данных, а не добавляются постфактум на этапе публикации релиза.

- требования безопасности фиксируются как проверяемые критерии приёмки для изменений;
- архитектурные решения (изоляция, привилегии, взаимодействие подсистем) принимаются с учётом модели угроз;
- на этапе проектирования определяется набор обязательных испытаний (SAST/фаззинг/DAST — по профилю компонента);
- несоответствие требованиям безопасности рассматривается как дефект, блокирующий выпуск.

4.2 Минимализм и предсказуемость

Минимальный состав и предсказуемое поведение базовой системы рассматриваются как базовый механизм снижения рисков. Каждый включаемый компонент увеличивает поверхность атаки, усложняет обновление и расширяет матрицу тестирования. Поэтому в НАЙС.ОС применяется принцип минимизации состава поставки и строгого контроля изменений.

- минимальный набор сервисов и пакетов в базовой установке;
- исключение необязательных сетевых слушателей по умолчанию;
- сокращение количества “вариантов состояния” системы за счёт фиксированных профилей и каналов обновлений;
- управляемые изменения: каждое изменение должно быть трассируемым (причина реализация проверка релиз).

4.3 Доказуемые сборки и прозрачность цепочки поставки

Цель “доказуемой сборки” — обеспечить возможность независимой проверки того, что опубликованные артефакты соответствуют заявленному исходному состоянию и процедурам сборки, а поставка защищена от подмены. Принцип включает

воспроизводимость, контроль источников, фиксацию метаданных и публикацию материалов для проверки.

- **воспроизводимость:** сборка выполняется в контролируемой среде; для релизов фиксируются версии инструментов и входные артефакты;
- **контроль исходников:** источники и патчи фиксируются по версиям/хэшам, изменения документируются и рецензируются;
- **SBOM:** формируется и публикуется состав компонентов (библиотеки, зависимости, версии) для релиза/образа/контейнерного слоя;
- **подпись и целостность:** пакеты/репозитории/образы сопровождаются средствами проверки подлинности и целостности.

Практический критерий “прозрачной поставки”
Для каждого релиза должен существовать минимальный проверяемый набор: (1) идентификатор версии/сборки, (2) список входных источников и патчей, (3) результаты обязательных проверок, (4) подписанные артефакты и материалы проверки (хэши/ключи/инструкции), (5) SBOM или эквивалентный перечень состава компонентов.

4.4 Безопасные значения по умолчанию (secure defaults)

Конфигурации по умолчанию в НАЙС.ОС выбираются с приоритетом безопасного состояния. Цель — обеспечить работоспособность в типовых сценариях, не создавая “неявных” сетевых экспозиций и не требуя от администратора немедленной ручной донастройки для базовой защиты.

- **сетевые политики по умолчанию:** ограничение входящих соединений и минимально необходимый набор разрешений;
- **минимально открытые сервисы:** отсутствие необоснованных слушающих портов и автозапуска дополнительных служб;
- **минимальные привилегии:** сервисы запускаются с ограниченными правами, где это поддерживается компонентами;
- **контроль конфигураций:** параметры безопасности документируются; отклонения фиксируются как управляемые изменения.

Замечание о совместимости
Безопасные значения по умолчанию могут требовать явной настройки для отдельных сценариев (например, маршрутизация/форвардинг, публикация сервисов,

расширение набора разрешённых портов). Такие изменения выполняются администратором как управляемая конфигурация и должны сопровождаться оценкой влияния на поверхность атаки и модель угроз.

4.5 Российская специфика: криптография и требования заказчиков (инженерная интерпретация)

В ряде проектов заказчики предъявляют требования к применяемым криптографическим механизмам, совместимости с отечественными алгоритмами, а также к форме предоставления подтверждающих материалов (документация, результаты испытаний, порядок обновления, контроль поставки). В НАЙС.ОС такие требования рассматриваются как дополнительные проверяемые требования безопасности и учитываются в жизненном цикле.

- **криптографическая совместимость:** обеспечение поддержки требуемых режимов и алгоритмов на уровне библиотек и приложений (при наличии соответствующих модулей/провайдеров);
- **управление конфигурацией криптопрофилей:** фиксация допустимых параметров, протоколов, наборов шифров и политик;
- **проверяемость поставки:** обязательность контроля целостности/подлинности обновлений и воспроизводимость артефактов релиза;
- **аудиторские артефакты:** комплектность материалов, необходимых для внутренней проверки заказчика (описание процессов, результаты испытаний, релизные протоколы, политика обновлений).

Результат применения принципов
Принципы Secure-by-design, минимализма, доказуемых сборок и secure defaults обеспечивают основу для управляемого риска: сокращение поверхности атаки, повышение предсказуемости релизов, проверяемость цепочки поставки и воспроизводимое устранение дефектов безопасности в рамках регламентированной поддержки.

5. Организация процесса: роли и ответственность

GOVERNANCE

Раздел определяет роли, распределение ответственности и порядок принятия решений в процессе безопасной разработки НАЙС.ОС. Ролевое разграничение используется для управления изменениями, обеспечения независимой проверки

(review), контроля выпуска (release) и организации реакции на уязвимости. Для каждой роли задаются полномочия и типовые обязанности, а также точки участия в жизненном цикле (требования реализация проверки релиз поддержка безопасности).

Принцип ответственности
Ответственность распределяется по принципу разделения полномочий: автор изменения не утверждает критические решения единолично; выпуск релиза и операции с ключами выполняются по отдельным правилам доступа, а результаты проверок фиксируются как артефакты, доступные для аудита.

5.1 Роли

Роль	Зона ответственности	Ключевые результаты/артефакты
Владелец продукта (Product Owner)	Определяет область действия продукта и целевые свойства релиза (функциональность, совместимость, ограничения). Утверждает приоритеты изменений, политику поддержки версий и допустимые риски.	План релиза, политика поддержки, решения по принятию рисков (если допускаются), утверждение состава релиза.
Владелец безопасности (Security Owner)	Отвечает за требования безопасности, модель угроз, критерии допуска релиза по безопасности, контроль выполнения обязательных проверок и качество процессов управления уязвимостями. Имеет право блокировать релиз при несоответствии требованиям безопасности.	Требования безопасности, модель угроз/актуализации, критерии gate-проверок, решения по критическим уязвимостям/исправлениям.

Роль	Зона ответственности	Ключевые результаты/артефакты
Релиз-инженер (Release Engineer)	Организует сборку релиза, обеспечивает воспроизводимость сборки и комплектность артефактов, выполняет релизные проверки, управляет каналами обновлений и публикацией. Обеспечивает применение процедур подписи и публикации.	Релизный протокол/чек-лист, логи сборки и тестов, подписанные артефакты, публикация в репозитории/каналы обновлений.
Мейнтейнер пакета (Package Maintainer)	Поддерживает пакет/подсистему: обновления версий, патчи, спецификации сборки, тесты, устранение дефектов. Обеспечивает соответствие правилам упаковки и требованиям безопасности (hardening, конфигурации по умолчанию, зависимостям).	Изменения в spec/патчах, результаты локальных проверок, описание изменений (changelog), задачи на исправления.
Рецензент (Reviewer)	Выполняет независимое рассмотрение изменений: корректность, безопасность, совместимость, соответствие требованиям и стандартам проекта. Проверяет полноту тестирования и результатов анализов (SAST/фаззинг/DAST — по профилю компонента).	Результат review (approve/changes requested), замечания, подтверждение критериев допуска (gate).

Роль	Зона ответственности	Ключевые результаты/артефакты
Команда реагирования на уязвимости (PSIRT/VDP)	Принимает сообщения об уязвимостях, выполняет triage, организует исправления, координирует раскрытие, выпускает advisory, контролирует сроки устранения и доведение информации до пользователей.	Карточка уязвимости/тикет, оценка критичности, advisory, план/статус исправления, отчёт о выпуске обновления.

5.2 RACI-матрица

В таблице используются обозначения: **R** — выполняет (Responsible), **A** — утверждает/несёт конечную ответственность (Accountable), **C** — консультирует/участвует (Consulted), **I** — получает уведомления (Informed).

Деятельность / решение	Product Owner	Security Owner	Release Eng.	Maintainer	Reviewer	PSIRT/VDP
Определение области действия продукта (scope) и состава релиза	A	C	R	C	I	I
Определение требований безопасности и критериев допуска релиза (security gates)	C	A	R	C	C	I
Анализ угроз / актуализация модели угроз	I	A	C	R	C	I
Разработка/изменение пакета (spec/патчи/конфигурации)	I	C	C	R	A	I
Рецензирование изменений (код/спеки/пакеты) и решение о merge	I	C	I	R	A	I
Выполнение обязательных проверок (SAST/фаззинг/DAST по профилю)	I	A	R	R	C	I

Деятельность / решение	Product Owner	Security Owner	Release Eng.	Maintainer	Reviewer	PSIRT/VDP
Сборка и выпуск релиза (freeze, build, sign, publish)	I	C	A	R	C	I
Управление ключами подписи (доступ, ротация, отзыв)	I	A	R	I	I	I
Приём сообщений об уязвимостях и triage	I	C	I	C	I	A/R
Исправление уязвимости и выпуск security-обновления	I	A	R	R	C	C
Публикация advisory и уведомление пользователей	I	A	C	C	I	R

5.3 Правила критических изменений

Критическими считаются изменения, которые влияют на: поверхность атаки, механизмы аутентификации/авторизации, криптографию и TLS/SSH параметры, политики безопасности по умолчанию, цепочку поставки, механизмы обновления, подписи и ключи, а также изменения, устраняющие уязвимости высокой/критической важности.

Категория критического изменения	Требования к обработке	Минимальные полномочия
Изменения, влияющие на безопасность по умолчанию	Требуются: описание влияния на модель угроз и поверхность атаки; результаты обязательных тестов; отдельное указание в релизных заметках; при необходимости — миграционные инструкции.	Merge: Reviewer + Security Owner.

Категория критического изменения	Требования к обработке	Минимальные полномочия
Изменения криптографии (параметры, провайдеры, профили)	Требуются: обоснование параметров; проверка совместимости; тесты на установку/обновление; подтверждение требований заказчиков. Любая смена параметров по умолчанию рассматривается как breaking change.	Merge: Reviewer + Security Owner. Release: Release Engineer.
Изменения цепочки поставки и сборки	Требуются: подтверждение источников и хэшей; проверка воспроизводимости; обновление SBOM/метаданных релиза; протокол изменения сборочной инфраструктуры; независимый review.	Merge: Reviewer + Release Engineer + Security Owner.
Изменения механизмов подписи, ключей и публикации	Требуются: регламент операции; журналирование; проверка процедуры отзыва/ротации ключей; контроль доступа; проведение релизной проверки на тестовом канале до применения на stable/LTS.	Операции: Release Engineer (по регламенту). Утверждение: Security Owner.
Исправления уязвимостей высокой/критической важности	Требуются: отдельный тикет/карточка уязвимости; оценка затронутых версий; тест на отсутствие регрессии; выпуск security-обновления по ускоренной процедуре; подготовка advisory.	Координация: PSIRT/VDP. Утверждение: Security Owner.

Ограничения доступа к ключам подписи

Доступ к закрытым ключам подписи артефактов релиза предоставляется ограниченному кругу лиц по регламенту. Ключевые операции (подпись, ротация, отзыв) должны выполняться на выделенной инфраструктуре и сопровождаться регистрацией действий. Передача ключей мейнтейнерам пакетов и авторам изменений не допускается.

Результат применения ролевой модели

Ролевая модель и RACI-распределение обеспечивают: (1) разделение полномочий, (2)

независимое подтверждение критических решений, (3) управляемый выпуск релизов и обновлений, (4) воспроизводимую реакцию на уязвимости с фиксируемыми артефактами.

6. Жизненный цикл разработки: этапы и артефакты

SSDLC / БРПО

Раздел описывает жизненный цикл разработки НАЙС.ОС в формате `этап → выполняемые действия → подтверждающие материалы`. Подтверждение подразумевает наличие воспроизводимых артефактов и записей (документы, отчёты, логи, протоколы, тикеты), пригодных для аудита и внутреннего контроля.

Трассируемость
Для релиза поддерживается трассируемость `требование → изменение → результаты проверок → артефакты поставки`. Трассируемость обеспечивается ссылками между документами, задачами (issue tracker), коммитами/merge request и отчётами CI.

6.1 Требования (security requirements)

Цель этапа — формализовать требования безопасности как проверяемые критерии, определить источники требований, приоритеты и ограничения релиза, а также подготовить критерии приёмки по безопасности.

Параметр	Что выполняется	Чем подтверждается
Источники требований	Сбор требований безопасности из нормативных источников, требований заказчика, результатов анализа угроз, внутренних политик (secure defaults, supply chain), а также из истории инцидентов/уязвимостей.	Реестр источников требований; ссылки на нормативные документы; протокол согласования с заказчиком (при наличии).

Параметр	Что выполняется	Чем подтверждается
Формализация требований	Перевод требований в проверяемую форму: объект, условие, критерий, способ проверки, уровень критичности. Требования классифицируются по компонентам/подсистемам и по типу контроля (дизайн/код/конфигурация/испытания/поставка).	Документ `Security Requirements` (SRD); матрица трассируемости; перечень обязательных проверок (gates).
Критерии приёмки безопасности	Определение критериев допуска релиза по безопасности: пороги по дефектам, результаты SAST/fuzzing/DAST, требования к подписи и целостности артефактов, требования к безопасным значениям по умолчанию.	Раздел “Security Acceptance Criteria”; релизный чек-лист (шаблон); политика блокировки релиза при нарушении критериев.

6.2 Проектирование и архитектура

Цель этапа — определить архитектурные границы доверия, поверхность атаки и меры снижения рисков. Этап включает анализ угроз и подготовку архитектурных материалов, используемых как основа для испытаний и критериев допуска.

Архитектурные артефакты для строгих контуров
Для контуров с повышенными требованиями (включая КИИ) архитектурное описание подсистем и интерфейсов рассматривается как обязательный подтверждающий материал. Требования к БРПО/испытаниям/поддержке безопасности для ПО в КИИ фиксируются приказом ФСТЭК №239 (официальная публикация: pravo.gov.ru).

Параметр	Что выполняется	Чем подтверждается
Анализ угроз	Определение активов, нарушителей, сценариев атак, границ доверия, потоков данных, критичных интерфейсов и условий эксплуатации. По результатам формируется перечень рисков и мер снижения.	Документ “Threat Model”; диаграммы потоков/границ доверия; реестр рисков; протокол утверждения (Security Owner).

Параметр	Что выполняется	Чем подтверждается
Архитектура подсистем и интерфейсов	Описание подсистем, внешних и внутренних интерфейсов (API, порты, протоколы, форматы), точек расширения, привилегированных операций и путей обновления. Формирование перечня интерфейсов как объекта испытаний.	Архитектурный документ; инвентаризация интерфейсов; “Attack Surface Inventory”; связь с планом испытаний.
Снижение рисков (risk treatment)	Проектирование мер: изоляция процессов, минимальные привилегии, sandboxing (где применимо), политики доступа, ограничения конфигурации, контроль прав и контекстов выполнения.	“Security Design Decisions”; требования к конфигурациям; протокол design review; задачи на реализацию мер.

6.3 Реализация (код / пакетирование)

Цель этапа — реализовать изменения в коде и/или упаковке (spec/патчи) с соблюдением правил безопасной разработки, обеспечив трассируемость изменений, воспроизводимость сборки и управляемость зависимостей.

Параметр	Что выполняется	Чем подтверждается
Secure coding	Применение правил безопасного кодирования на уровне языка/компонента (валидация входных данных, безопасные API, управление памятью, обработка ошибок, исключение утечек секретов, корректное логирование).	Правила/гайдлайны проекта; результаты code review; отчёты SAST (как минимум на релизной ветке).
Policy для патчей к upstream	Любой патч должен быть трассируемым: причина, ссылка на issue/CVE (если применимо), описание влияния, наличие теста или воспроизводимого сценария проверки. Патчи минимизируются и сопровождаются планом снятия (если возможно).	Запись в issue tracker; ссылка в spec/патче; changelog; результаты тестов; связь “патч релиз”.

Параметр	Что выполняется	Чем подтверждается
RPM/spec в НАЙС.ОС	Управление зависимостями, включение флагов hardening (где применимо), контроль прав и пользователей, запрет сетевых “магических скачиваний” в build-time, запрет нефиксированных источников, фиксация хэшей/версий.	Spec-файл и правила упаковки; логи сборки; отчёт проверки “no network in build”; протокол review для критичных изменений.

Требование к управляемости зависимостей
Изменение зависимостей (добавление новых библиотек, смена крупных версий, изменение ABI/совместимости) считается изменением, влияющим на поверхность атаки и матрицу тестирования. Такие изменения требуют отдельной фиксации причины, оценки влияния и расширенного набора проверок.

6.4 Верификация безопасности (испытания)

Цель этапа — подтвердить выполнение требований безопасности и выявить дефекты до выпуска. Набор испытаний определяется профилем компонента и требованиями контура. Для ПО, применяемого в КИИ, класс испытаний включает SAST и fuzzing, а для наиболее строгих случаев — также DAST (приказ ФСТЭК №239: pravo.gov.ru).

Класс проверки	Что проверяется	Чем подтверждается
SAST	Статический анализ исходного кода/артефактов сборки по правилам безопасности. Используется для выявления классов дефектов по потокам данных, использованию API и опасным паттернам.	Отчёты SAST с версией правил/конфигурации; список выявлений; решения по каждому выявлению (исправлено/ложноположительное/принято с риском).
Fuzzing	Фаззинг входов компонента: протоколы, парсеры, форматы, API. Цель — выявление крашей, зависаний, дефектов обработки границ и потенциально эксплуатируемых ошибок.	Отчёты/логи фаззинга; наборы входов/корпус; подтверждение устранения найденных крашей; связка “кейс фикс регресс-тест”.
DAST	Динамическое тестирование на выполняющемся экземпляре: проверка реальных маршрутов/эндпоинтов/протоколов, конфигурации безопасности, сценариев аутентификации/авторизации и интеграционных эффектов.	Отчёты DAST (профиль, цели, результаты); протокол настройки стенда; подтверждение исправления выявлений.

Класс проверки	Что проверяется	Чем подтверждается
Unit/Integration	Функциональные тесты и интеграционные проверки, покрывающие требования безопасности (валидация, права, обработка ошибок, корректность протоколов, отказоустойчивость в допустимых пределах).	Результаты тестов в CI; отчёты покрытия (если применимо); протокол “tests passed” на релизной ветке.
Регресс security-тестов	Набор регрессионных проверок по ранее найденным дефектам и классам рисков (недопущение повторного появления).	Набор регресс-тестов; ссылки на инциденты/уязвимости; отчёты прохождения тестов на релизе.
Проверка конфигураций	Контроль конфигураций по базовым требованиям безопасности (CIS-подобная логика): сервисы, порты, права, политики, параметры протоколов и криптографии (по профилю поставки).	Отчёты конфигурационных проверок; список отклонений; решения (исправлено/допущено) и обоснование.
Тест обновления/rollback	Проверка сценариев обновления и отката (если применимы: OSTree/атомарные механизмы): корректность смены версии, сохранность критичных настроек, отсутствие деградации безопасных дефолтов.	Протокол испытаний обновления/rollback; логи выполнения; критерии успешности; фиксация результатов.

6.5 Релиз и поставка

Цель этапа — выполнить управляемый выпуск релиза: зафиксировать состав, завершить обязательные проверки, сформировать артефакты поставки и обеспечить проверяемую подлинность и целостность (подписи/хэши/ключи).

Элемент этапа	Что выполняется	Чем подтверждается
Freeze	Заморозка релизной ветки/состава; запрет некритичных изменений; фиксация версий, патчей и конфигураций.	Метка релиза; протокол freeze; список включённых изменений; ссылки на тикеты/merge request.

Элемент этапа	Что выполняется	Чем подтверждается
Обязательные проверки (gates)	Выполнение набора проверок, определённых в критериях приёмки безопасности: тесты, SAST/fuzz/DAST (по профилю), конфигурационные проверки, тест обновления (если применимо).	Отчёты CI; отчёты анализаторов; релизный чек-лист с отметками; решения по выявлениям.
Подписание	Подпись артефактов: пакеты, репозиторные метаданные, образы (ISO/RAW/OSTree/контейнерные), манифесты. Ключи подписи управляются по отдельному регламенту доступа.	Подписи (detached/inline по формату); публикация публичных ключей; инструкции проверки; логи операции подписи.
Публикация	Публикация в официальных каналах поставки и обновлений. Обновления попадают в stable/LTS по правилам, исключающим неаудированные изменения.	Репозиторные метаданные; записи публикации; правила каналов; протокол продвижения изменений (testing stable).
Релизные материалы	Формирование и публикация материалов релиза: release notes, список изменений, SBOM, инструкции проверки подписи/хэшей, информация о поддерживаемых версиях и политика обновлений.	Release notes; SBOM; манифесты; контрольные суммы; комплект релизной документации.

6.6 Поддержка безопасности после релиза

Цель этапа — обеспечить управляемое устранение уязвимостей и доставку исправлений с подтверждаемой подлинностью и целостностью, а также регламентированное уведомление пользователей. Требования к наличию процедур, уведомлений и проверяемой поставки обновлений для ПО в КИИ зафиксированы приказом ФСТЭК №239 (pravo.gov.ru).

Элемент этапа	Что выполняется	Чем подтверждается
SLA по уязвимостям	Установка целевых сроков обработки в зависимости от критичности (triage, исправление, выпуск обновления, уведомление). SLA фиксируется для поддерживаемых веток (stable/LTS).	Политика SLA; матрица “критичность срок”; отчёты соблюдения сроков по выборке инцидентов.
Приём и triage	Приём сообщений, регистрация, воспроизведение, определение затронутых версий/конфигураций, оценка критичности, определение мер снижения риска до выпуска исправления.	Тикет уязвимости; протокол воспроизведения; оценка критичности; список затронутых версий; решение о плане исправления.
Исправление и backport	Подготовка патча, тестирование, выпуск обновления. При необходимости выполняется backport исправления в поддерживаемые LTS-ветки с контролем совместимости и регресса.	Коммиты/merge request; результаты тестов; протокол backport; релизные заметки для security-обновления.
Advisory	Публикация уведомления о безопасности. Минимальный состав: описание, затронутые версии, влияние, компенсирующие меры (mitigation), наличие исправления, порядок обновления, материалы проверки целостности/подлинности обновления.	Опубликованный advisory; ссылка на пакеты/обновления; идентификаторы исправлений; контрольные суммы/подписи.
Доставка обновлений	Публикация исправления в каналах обновлений; обеспечение проверки подписи/хэшей на стороне потребителя; фиксация завершения поставки и контроль доступности артефактов.	Подписанные пакеты/репозитории; инструкции проверки; логи публикации; журнал изменений канала.

Выходные артефакты жизненного цикла

Для каждого релиза и security-обновления формируется минимальный комплект: (1) перечень требований/критериев допуска, (2) архитектурные и threat-model артефакты (по применимости), (3) отчёты испытаний и решения по выявлениям, (4) релизный

протокол/чек-лист, (5) подписанные артефакты поставки и материалы проверки (ключи/хэши/инструкции), (6) SBOM (по политике), (7) для уязвимостей — тикет/оценка/патч/advisory/сведения о доставке обновления.

7. Безопасность цепочки поставки (supply chain)

SUPPLY CHAIN

Раздел описывает меры безопасности цепочки поставки НАЙС.ОС: от получения исходников и патчей до сборки, формирования SBOM, подписи артефактов и публикации обновлений. Цель — исключить неучтённые изменения, обеспечить воспроизводимость сборки и предоставить потребителю проверяемые механизмы подтверждения целостности и подлинности поставляемых артефактов.

Цель контроля цепочки поставки
Контроль цепочки поставки направлен на обеспечение трёх свойств: **(1) идентифицируемость** входов (источники/версии/хэши), **(2) воспроизводимость** сборки (одинаковый результат при одинаковых входах), **(3) проверяемость** выпуска (подписи/метаданные/материалы проверки).

7.1 Управление исходниками и патчами

Источники исходного кода и патчей рассматриваются как критический вход сборки. Для каждого пакета фиксируются происхождение, версия, целостность и правила получения. Допускаются только управляемые источники, обеспечивающие воспроизводимое получение одинакового содержимого.

Контроль	Как выполняется	Чем подтверждается
Происхождение исходников	Для каждого пакета фиксируется upstream-источник (репозиторий/релизный архив), политика зеркалирования и правила обновления версии. Приоритет отдаётся официальным релизам и проверяемым каналам распространения.	Спецификация источника в spec/манифесте; ссылка на upstream; запись в changelog/issue tracker.

Контроль	Как выполняется	Чем подтверждается
Фиксация версий и хэшей	Входные артефакты фиксируются по версии и контрольным суммам (хэшам). Изменение источника или версии оформляется как управляемое изменение с оценкой влияния на зависимости и проверки.	Хэши/метаданные в репозитории спецификаций; протокол обновления версии; diff входов.
Зеркала и доступность	Используются контролируемые зеркала/кэши исходников для снижения рисков подмены, недоступности и “дрейфа” содержимого. Политика зеркалирования должна обеспечивать повторяемое получение одного и того же контента.	Настройки зеркал; журнал синхронизации; перечень источников и зеркал (inventory).
Управление патчами	Патчи к upstream допускаются только при наличии трассируемости: причина, ссылка на issue/CVE (если применимо), описание влияния и подтверждение тестом/сценарием проверки. Патчи минимизируются и подлежат review.	Patch-файлы в репозитории; ссылки на тикеты; результаты тестов; review-решения.

7.2 “Никаких сюрпризов на сборке” (изоляция build-time)

Сборка пакетов и образов должна быть детерминированной по входам. В build-time запрещаются неучтённые сетевые обращения, скачивания зависимостей и генерация артефактов из внешних источников, не зафиксированных в спецификациях. Любая попытка “дотянуть” зависимости из сети рассматривается как нарушение цепочки поставки.

Запрет сетевых зависимостей в build-time
Сборочная среда должна исключать неуправляемые сетевые обращения в процессе сборки (download-on-build). Допускаются только источники, заранее зафиксированные и проверенные (зеркала исходников/репозитории зависимостей), а способ их использования фиксируется регламентом сборки.

Контроль	Как выполняется	Чем подтверждается
Изоляция сборочной среды	Сборка выполняется в контролируемой среде (изолированный агент/контейнер/чрут) с предопределённым набором инструментов, с фиксированными зависимостями сборки и политиками доступа к сети.	Описание build environment; версии toolchain; логи сборки; конфигурация CI.
Контроль сетевых обращений	Используются технические ограничения (политики сети/ACL) и/или проверки, подтверждающие отсутствие сетевых скачиваний во время сборки. Нарушение блокирует сборку/релиз.	Отчёт “no network in build”; логи сетевой политики; результаты контрольной сборки.
Фиксация зависимостей сборки	BuildRequires и инструментальные зависимости фиксируются и проходят обновление как управляемое изменение (оценка влияния на результаты сборки, тестирование, воспроизводимость).	Spec/manifest; протокол обновления зависимостей; результаты пересборки контрольного набора пакетов.

7.3 SBOM (Software Bill of Materials)

SBOM используется как машиночитаемое описание состава релиза: какие пакеты, библиотеки и версии входят в поставку. SBOM обеспечивает прозрачность состава и упрощает анализ влияния уязвимостей (impact analysis) для потребителя и производителя. SBOM связывается с конкретным релизом и артефактами поставки (репозиторий, образы, контейнерные слои).

Контроль	Как выполняется	Чем подтверждается
Формирование SBOM	SBOM формируется автоматически в процессе сборки/релиза на основании фактического состава пакетов/слоёв. Формат и состав полей фиксируются политикой (минимально: имя, версия, источник/поставщик, идентификаторы, хэши по возможности).	Файл SBOM (формат по политике); лог генерации; версия инструмента формирования SBOM.

Контроль	Как выполняется	Чем подтверждается
Связь SBOM релиз	SBOM привязывается к идентификатору релиза/сборки и к манифесту артефактов поставки. Для релиза фиксируется набор артефактов, к которым относится SBOM (репо/ISO/RAW/OSTree/контейнер).	Release manifest; ссылки в release notes; метаданные релиза; контрольные суммы SBOM.
Связь SBOM пакет	Для пакетов фиксируются идентификаторы сборок и версии; SBOM позволяет сопоставить пакет/версию с релизом. При обновлениях SBOM обновляется одновременно с публикацией артефактов.	Репозиторные метаданные; списки пакетов релиза; соответствие “package NEVRA SBOM entry”.
Публикация SBOM	SBOM публикуется в составе релизных материалов и должен быть доступен для скачивания по официальному каналу. Для SBOM применяются механизмы целостности/подлинности (подпись/хэш).	Ссылка на опубликованный SBOM; подпись/хэш; инструкция проверки.

7.4 Reproducible builds (воспроизводимые сборки)

Воспроизводимая сборка означает возможность получить идентичные артефакты при одинаковых входах и одинаковой сборочной среде. В НАЙС.ОС воспроизводимость применяется как контроль целостности процесса: она снижает риски скрытой подмены и упрощает аудит.

Контроль	Что проверяется	Чем подтверждается
Детерминизм входов	Фиксация исходников/патчей/инструментов; исключение недетерминизма (время сборки, порядок файлов, случайные значения) там, где это возможно на уровне пакета.	Манифест входов; зафиксированные версии toolchain; результаты сравнения артефактов (diff/хэши).

Контроль	Что проверяется	Чем подтверждается
Контрольные пересборки	Выполняются контрольные пересборки выборки пакетов/образов: (1) на релизной ветке, (2) периодически по расписанию (частота фиксируется политикой), (3) после изменения toolchain/сборочной среды. Несовпадения анализируются и оформляются как дефекты процесса/пакета.	Отчёты пересборок; список проверенных пакетов; протокол разбора расхождений; исправления в spec/параметрах сборки.
Порог допустимых расхождений	Политикой определяются допустимые отклонения (если применимо) и способы их устранения. Для критичных компонентов приоритет — достижение полного совпадения артефактов.	Политика reproducible builds; протокол решений; результаты устранения недетерминизма.

7.5 Подпись артефактов и управление ключами

Подпись артефактов используется для подтверждения подлинности (кто выпустил) и целостности (что не изменено) поставки. Управление ключами подписи рассматривается как отдельный критический процесс, требующий ограничений доступа, регламента операций, ротации и процедуры отзыва ключей.

Ключи подписи как критический актив
Закрытые ключи подписи не являются частью обычного процесса разработки и недоступны авторам изменений и мейнтейнерам пакетов по умолчанию. Операции подписи выполняются на выделенной инфраструктуре и подлежат регистрации. При компрометации ключа требуется процедура отзыва и выпуск обновлённых материалов доверия (публичные ключи/инструкции).

Контроль	Как выполняется	Чем подтверждается
Подпись пакетов и репозиториев	Пакеты и/или репозиторные метаданные подписываются ключом релиза. Проверка подписи выполняется на стороне потребителя средствами менеджера пакетов/обновлений.	Подписи; публикация публичного ключа; инструкция проверки; протокол подписи и публикации.

Контроль	Как выполняется	Чем подтверждается
Подпись образов/слоёв	Образы (ISO/RAW/OSTree/контейнерные) сопровождаются подписью и/или контрольными суммами, опубликованными по официальному каналу. Манифест артефактов включает хэши и идентификаторы версий.	Release manifest; подписи/хэши; публичный ключ; релизные заметки с ссылками на материалы проверки.
Хранение ключей	Ключи хранятся в защищённом виде. Допускаются варианты: HSM, офлайн-ключи, выделенные хранилища с MFA и аппаратной защитой. Выбранный вариант фиксируется политикой и регламентом операций.	Политика хранения ключей; журнал доступа; описание инфраструктуры подписи.
Ротация и отзыв	Определяются правила ротации ключей (плановая/внеплановая), условия отзыва, порядок перевыпуска подписей и обновления доверенных ключей на стороне потребителя.	Регламент ротации/отзыва; опубликованные уведомления (при необходимости); новая цепочка доверия и инструкции миграции.

Выходные результаты блока supply chain
Контроль цепочки поставки обеспечивает: (1) фиксируемые и проверяемые входы сборки (источники/версии/хэши), (2) отсутствие сетевых “сюрпризов” при сборке, (3) прозрачность состава релиза через SBOM, (4) воспроизводимость (через контрольные пересборки), (5) проверяемую подлинность и целостность поставки через подпись и управление ключами.

8. Защита среды разработки и сборки

BUILD / CI SECURITY

Раздел фиксирует меры защиты среды разработки и сборки НАЙС.ОС как критического элемента процесса безопасной разработки. Среда разработки и сборки рассматривается как часть цепочки поставки: компрометация этой среды приводит к рискам внедрения несанкционированных изменений в артефакты поставки даже при корректном исходном коде.

Цель защиты среды
Целью является обеспечение проверяемых свойств: **(1) контролируемый доступ** к исходникам и инфраструктуре, **(2) воспроизводимость и неизменяемость** сборочной среды, **(3) наблюдаемость** действий (журналирование и расследование), **(4) устойчивость** к типовым сценариям компрометации сборочной цепочки.

8.1 Контроль доступа (MFA, least privilege, журналирование)

Контроль доступа применяется ко всем системам разработки и сборки: репозиториям исходников, системе задач, CI/CD, реестрам артефактов, инфраструктуре подписи и публикации. Доступ предоставляется по принципу минимально необходимых прав и подлежит регистрациям событий безопасности.

Контроль	Как реализуется	Чем подтверждается
MFA и усиленная аутентификация	MFA включается для административных и привилегированных операций: управление CI, доступ к артефактам релиза, настройка репозиториев, операции публикации и управления ключами. Для сервисных учётных записей применяются отдельные токены с ограниченными правами и сроком действия.	Политики доступа; конфигурации IdP/SSO (при наличии); журналы входа; перечень привилегированных ролей.
Least privilege и разграничение ролей	Права назначаются по ролям (разработка, review, релиз, безопасность) и ограничиваются по операциям. Запрещается совмещать права на изменение исходников и права на подпись/публикацию релиза в одной роли по умолчанию.	RBAC-матрица; списки групп и ролей; протоколы выдачи/отзыва доступа; результаты периодического пересмотра прав.

Контроль	Как реализуется	Чем подтверждается
Защита веток и контроль merge	Ветки релиза/основные ветки защищены: запрет прямых пушей, обязательный review, обязательное прохождение CI-проверок, запрет обхода проверок без отдельного полномочия. Критические изменения требуют участия Security Owner и/или Release Engineer.	Настройки защищённых веток; правила merge; журналы merge request; подтверждение прохождения gate-проверок.
Журналирование и аудит действий	Журналируются: аутентификация, выдача прав, операции в репозиториях, изменения CI-конфигураций, публикации артефактов, операции с ключами подписи. Журналы защищаются от модификации и доступны для расследования.	Аудит-логи; политики хранения; отчёты выборочного аудита; следы корреляции “изменение сборка публикация”.

Требование к привилегированным операциям
Привилегированные операции (управление CI, публикация релиза, операции подписи, изменение политик доступа) выполняются только из доверенных контуров доступа и с обязательной фиксацией: кто, когда, что изменил и по какому основанию. Любой обход контрольных процедур рассматривается как инцидент безопасности процесса.

8.2 Изоляция билд-агентов и неизменяемые образы сборки

Сборка и тестирование выполняются на изолированных билд-агентах. Сборочная среда должна быть предсказуемой: образ/контейнер сборки формируется централизованно, фиксируется по версии/хэшу и не изменяется “на месте”. Это снижает риск дрейфа среды и усложняет внедрение скрытых изменений.

Контроль	Как реализуется	Чем подтверждается
Эфемерные (одноразовые) агенты	Агент создаётся под задачу сборки/тестов и уничтожается после выполнения. Долгоживущие агенты допускаются только при наличии строгих мер контроля целостности и регулярной переинициализации.	Конфигурация CI; журналы запуска агентов; политика времени жизни; отчёты о переинициализации.
Сетевые ограничения build-time	Сетевой доступ агентов минимизируется: запрет исходящих соединений по умолчанию либо разрешение только на контролируемые источники (зеркала, реестры артефактов). Это предотвращает “скачивание по требованию” и утечки.	Политики сети/ACL; отчёты “no network in build” (по политике); логи сетевых блокировок.
Неизменяемые (immutable) образы сборки	Среда сборки поставляется как неизменяемый образ, фиксируемый по версии и хэшу/дигесту. Обновление образа оформляется как отдельное изменение с review и контрольной пересборкой выборки пакетов.	Реестр образов с версиями/дигестами; протокол обновления; результаты контрольной пересборки; журнал распространения образов.
Секреты и токены в CI	Секреты не включаются в исходники/образы. Доступ к секретам ограничивается задачами и окружениями, применяется минимальный срок жизни токенов и запрет на вывод секретов в логи.	Конфигурация секрет-хранилища; политики выдачи токенов; проверка утечек (secret scanning); журнал доступа к секретам.

8.3 Предотвращение компрометации сборочной цепочки

Ниже приведены типовые сценарии атак на сборочную цепочку и меры, применяемые для их предотвращения или снижения последствий. Меры ориентированы на практическую проверяемость: для каждой меры предусмотрен

подтверждающий артефакт (настройка, лог, протокол, отчёт).

Сценарий риска	Меры предотвращения / снижения	Чем подтверждается
Внедрение изменений в исходники (compromised account / bypass review)	MFA; защита веток; обязательный review; обязательные CI-gates; запрет прямых пушей; журналирование и расследование по цепочке “учётная запись MR сборка артефакт”.	Настройки защищённых веток; журналы MR/approve; отчёты CI; аудит-логи репозитория.
Подмена исходников/зависимостей при получении	Фиксация версий/хэшей; использование контролируемых зеркал; запрет нефиксированных источников; проверка целостности входов перед сборкой.	Манифест источников; хэши; журналы синхронизации зеркал; логи проверки входов.
Выполнение произвольного кода в CI (компрометация runner/agent)	Эфемерные агенты; минимальные права; сегментация сети; запрет доступа к ключам подписи из CI; контроль целостности образов сборки; ограничение секретов по окружениям.	Конфигурация CI runners; политики сети; журналы выдачи секретов; перечень разрешённых операций агента.
Утечка секретов (токены, ключи, пароли)	Secret scanning; запрет секретов в исходниках; токены с минимальным сроком жизни; раздельные секреты по средам; маскирование в логах; обязательная ротация при инцидентах.	Отчёты secret scanning; политика ротации; журналы доступа к секретам; инцидентные протоколы (при наличии случаев).
Несанкционированная публикация артефактов / подмена репозитория	Разделение полномочий публикации; обязательная подпись пакетов/метаданных/образов; ограничение прав на публикацию; неизменяемое хранение релизных манифестов (WORM-подобная модель хранения по политике).	Журнал публикаций; подписи артефактов; release manifest; RBAC для репозитория артефактов.

Сценарий риска	Меры предотвращения / снижения	Чем подтверждается
Несанкционированное использование ключей подписи	Выделенный контур подписи; хранение ключей в HSM или офлайн-формате (по политике); ограничение доступа; регистрация операций; ротация и отзыв ключей по регламенту.	Регламент управления ключами; журналы операций подписи; материалы ротации/отзыва; перечень доверенных публичных ключей.

Требование “ключи подписи вне CI”
Закрытые ключи подписи артефактов релиза не размещаются на билд-агентах и не доступны из общего контура CI. Подпись релизных артефактов выполняется в отдельном контуре и по регламенту, обеспечивающему контроль доступа, регистрацию операций и возможность отзыва ключей.

8.4 Инструментальный контроль процесса (контекст сертификации)

Защита среды разработки и сборки рассматривается как проверяемый элемент процесса безопасной разработки. Для сертификации процессов ожидается возможность подтвердить не только наличие регламентов, но и фактическую реализацию процедур и технических контролей (включая контроль среды разработки и сборки) средствами инструментального контроля.

Практический смысл “инструментального контроля”
Под инструментальным контролем в настоящем документе понимается проверка фактической реализации процесса на основе технических свидетельств: конфигураций систем, журналов, протоколов, результатов CI, артефактов сборки и публикации, а также выборочного воспроизведения операций (например, контрольная сборка, проверка подписи, проверка политики доступа).

Объект проверки	Что должно подтверждаться	Примеры подтверждающих материалов
Системы контроля версий	Защита веток; обязательность review; журналы изменений; управление правами; неизменяемость релизных меток/тегов.	Конфигурации репозитория; audit logs; правила merge; выгрузки настроек защищённых веток.
CI/CD и билд-агенты	Изоляция агентов; неизменяемость образов; сетевые ограничения build-time; управление секретами; воспроизводимость задач.	Конфигурации runners; реестр образов (версии/дигесты); логи CI; отчёты “no network in build” (по политике).
Артефакты сборки и публикации	Трассируемость “изменение сборка артефакт публикация”; целостность и подлинность; неизменяемость релизного набора.	Release manifest; подписи/хэши; журналы публикации; метаданные репозиториев.
Контур подписи и ключевая инфраструктура	Разделение контуров; ограничения доступа; регистрация операций; процедуры ротации и отзыва; отсутствие ключей в CI.	Регламенты; журналы подписи; перечни доверенных ключей; протоколы ротации/отзыва; описание хранения (HSM/офлайн).

Требование к аудиторской воспроизводимости
Процесс считается аудиторски воспроизводимым только при наличии непротиворечивой цепочки доказательств: (1) утверждённые регламенты и роли, (2) технические настройки, реализующие регламенты, (3) логи и артефакты, подтверждающие фактическое выполнение, (4) возможность выборочного повторения контрольных операций (контрольная сборка, проверка подписи, проверка политики доступа) на ограниченном наборе объектов.

9. Управление уязвимостями и взаимодействие с БДУ/исследователями

VDP / PSIRT / БДУ

Раздел описывает порядок приёма сообщений об уязвимостях, triage (подтверждение

и оценка), согласованное раскрытие (coordinated disclosure), выпуск исправлений и подготовку данных для взаимодействия с БДУ и внешними исследователями. Процесс ориентирован на воспроизводимость и аудит: каждое обращение фиксируется, имеет статус, владельца, набор подтверждений и результаты устранения.

Базовые принципы
1) конфиденциальность до согласованного раскрытия; 2) воспроизводимость подтверждения; 3) прозрачность для потребителей через advisory и обновления; 4) трассируемость `сообщение → triage → исправление → релиз/обновление → advisory`.

9.1 Каналы приёма сообщений

Для обеспечения конфиденциальности и полноты исходных данных используются выделенные каналы приёма. Публичные каналы поддержки не предназначены для передачи PoC/эксплойтов и технических деталей до согласования.

Канал	Назначение	Минимальные требования к сообщению
security@niceos.ru	Основной канал для приватных сообщений об уязвимостях (VDP/PSIRT). Используется для первичного контакта, обмена PoC, согласования сроков раскрытия и передачи патчей/обходных мер до публикации.	Описание, затронутый продукт/версия, условия воспроизведения, ожидаемое/фактическое поведение, контакт.
PGP (шифрование)	Для передачи чувствительных материалов допускается шифрование/подпись PGP. Публичные ключи и правила использования публикуются в политике раскрытия уязвимостей (VDP) НАЙС.ОС.	PGP-ключ; подпись сообщения; идентификатор ключа и отпечаток (fingerprint).

Канал	Назначение	Минимальные требования к сообщению
Приватный тикет в баг-трекере	Используется для работы с заказчиками/партнёрами и для внутренних компонентов, когда требуется контроль статусов, исполнителей и связей с релизами. Тикет должен быть закрыт от публичного доступа до завершения согласованного раскрытия.	Репродукция, артефакты (логи/дампы), затронутые версии, оценка влияния.
БДУ (при необходимости)	Для передачи сведений оператору БДУ используются каналы, предусмотренные регламентом БДУ (включая веб-форму/почту и PGP). Данный канал применяется при наличии оснований для включения сведений в БДУ и при соблюдении режима раскрытия.	Набор полей по регламенту БДУ (см. 9.4) + контактные данные.

Ограничения на эксплуатацию PoC
PoC/эксплойты используются только в контролируемой лабораторной среде для подтверждения уязвимости и проверки исправления. Действия, не необходимые для подтверждения, не выполняются. Материалы PoC не публикуются до согласованного раскрытия.

9.2 Triage: подтверждение, критичность, затронутые версии, решение о CVE

Triage выполняется для перевода сообщения в инженерно проверяемую задачу: воспроизведение, локализация причины, определение затронутых версий/профилей поставки, оценка критичности и выбор стратегии исправления (upstream/backport/mitigation).

Шаг	Что выполняется	Результат (фиксируемые данные)
Регистрация	Присвоение внутреннего идентификатора, фиксация источника сообщения, конфиденциальности, контактов и исходных материалов.	Тикет/запись обращения; владелец; статус; метки продукта/компонента.
Воспроизведение	Подготовка стенда, повторение шагов исследователя, сбор подтверждений (логи, трейсы, дампы), минимизация условия воспроизведения.	Протокол воспроизведения; артефакты подтверждения; “reproducible / not reproducible”.
Оценка критичности	Расчёт CVSS (вектор и балл) и качественная оценка влияния (C/I/A), наличие предпосылок эксплуатации, доступность обходных мер.	CVSS-вектор; уровень опасности; обоснование; приоритет устранения.
Затронутые версии	Определение диапазона затронутых версий (релизы НАЙС.ОС, пакеты, профили поставки), условий эксплуатации и конфигураций, влияющих на наличие/эксплуатацию уязвимости.	“Affected versions”; “Fixed in”; условия/конфигурации; перечень веток для backport.
Решение о CVE	Определение необходимости CVE и пути получения: координация с upstream/вендором, использование существующего CVE, либо инициирование запроса через соответствующие каналы (если применимо). Для distro-specific исправлений допускается выпуск advisory с внутренним идентификатором при отсутствии CVE.	“CVE: existing / requested / not applicable”; ссылка на upstream; решение о формате идентификаторов.

9.3 Coordinated disclosure: правила и сроки

Согласованное раскрытие применяется для минимизации риска массовой эксплуатации до выхода исправления. Взаимодействие с исследователями строится на фиксируемых сроках ответа, промежуточных статусах и согласовании даты публикации advisory.

Этап раскрытия	Правило	Выходной артефакт
Подтверждение получения	Сообщение регистрируется, исследователь получает подтверждение получения и первичный идентификатор обращения.	Ответ исследователю; внутренний ID; режим раскрытия (private).
Согласование таймлайна	Фиксируется план: дата целевого исправления, условия публикации (после исправления/после выхода обновления), формат упоминания исследователя (по согласованию).	Зафиксированный таймлайн; статусные уведомления.
Публикация advisory	Публикация выполняется после выпуска исправления (или одновременно с ним). Advisory содержит затронутые версии, влияние, меры снижения риска, порядок обновления и материалы проверки целостности/подлинности обновлений.	Security advisory; ссылки на обновления; “Fixed in”.
Исключения	При наличии сведений о активной эксплуатации допускается ускоренный выпуск обновлений и более ранняя публикация краткого предупреждения с компенсирующими мерами до выхода полного исправления.	Early advisory / mitigation notice; статус инцидента.

Ограничение на публичное раскрытие до согласования
До согласованного раскрытия не публикуются PoC/детали эксплуатации, позволяющие воспроизвести атаку на незащищённые инсталляции. При необходимости внешнего раскрытия (например, из-за утечки в публичных источниках) режим раскрытия пересматривается и фиксируется отдельным решением.

9.4 Взаимодействие с БДУ: состав сведений и сроки (по регламенту)

При подготовке сведений для БДУ используется структура и состав полей, определённые регламентом включения информации об уязвимостях. Практически это означает, что материалы triage должны быть оформлены так, чтобы их можно было преобразовать в описание уязвимости для публикации и сопровождения (включая идентификаторы, вектор, затронутые версии и подтверждающие материалы).

Поле (минимальный набор)	**Содержание**	**Источник данных в процессе**
Наименование и описание	Краткое наименование и техническое описание сущности дефекта и условий эксплуатации.	Результаты triage; протокол воспроизведения; анализ причины.
ПО/ПАС и версии	Наименование уязвимого ПО/ПАС и диапазоны затронутых версий, включая “Fixed in”.	“Affected versions”; пакет/релиз; матрица веток и backport.
Вендор	Производитель/разработчик (при наличии), включая разграничение upstream и поставки НАЙС.ОС (если релевантно).	Метаданные пакета; upstream-ссылки; сведения от производителя.
CWE (тип/идентификатор ошибки)	Тип ошибки и идентификатор по CWE (Common Weakness Enumeration).	Анализ причины; результаты SAST/ревью; классификация дефекта.
Платформы и ОС	ОС и аппаратные платформы, для которых актуальна уязвимость (если применимо).	Матрица сборок/архитектур; протокол воспроизведения.
CVSS (вектор и степень опасности)	Базовый вектор и оценка опасности по CVSS (вектор и балл). Уровень опасности фиксируется на основании балла.	Расчёт CVSS; обоснование параметров; приоритизация устранения.

Поле (минимальный набор)	Содержание	Источник данных в процессе
Проверка и подтверждающие материалы	Порядок проверки уязвимости и подтверждения: PoC-код, видеодемонстрация или иные материалы.	Набор PoC; шаги воспроизведения; артефакты подтверждения.
Контактные данные	Контакты изготовителя/исследователя для уточнений и согласования описания/раскрытия.	Данные обращения; служба безопасности/PSIRT.
Идентификаторы в иных системах	CVE/другие идентификаторы (если присвоены), ссылки на публикации и статус уязвимости/исправления.	Upstream advisory; CVE; запись в трекере; release notes.

Сроки взаимодействия с БДУ зависят от роли участника (изготовитель/исследователь/оператор). Для изготовителя регламент фиксирует, что сведения о выявленной уязвимости направляются в БДУ в течение **3 рабочих дней** с даты выявления. Для оператора БДУ регламент фиксирует сроки размещения согласованного описания в Банке данных угроз: не позднее **5 рабочих дней** для уязвимостей критического/высокого уровня опасности и не позднее **7 рабочих дней** для среднего/низкого уровня с момента получения информации от изготовителя.

Практический вывод для процесса НАЙС.ОС
Чтобы взаимодействие с БДУ было технически “без доработок”, triage-пакет НАЙС.ОС формируется в структуре: описание затронутые версии CWE CVSS порядок проверки PoC/подтверждения меры устранения статусы/даты идентификаторы. Это упрощает согласование описания, выпуск advisory и подготовку корректных обновлений.

10. Как проверять НАЙС.ОС

AUDIT / VERIFY

Раздел задаёт минимально достаточные проверки для аудиторов и заказчиков: что именно следует запросить и как быстро убедиться, что релиз и обновления НАЙС.ОС поставляются проверяемо (подписи/хэши), прозрачно (SBOM) и с управляемой поддержкой безопасности (advisory, процедуры triage и сроки).

Базовый критерий доверия
Проверка считается пройденной только при наличии связанной цепочки свидетельств: **релиз манифест артефактов подписи/хэши SBOM политика обновлений advisory**. Отсутствие любого звена делает оценку неполной.

10.1 Быстрый чек-лист (10 минут)

Чек-лист предназначен для первичной проверки релиза без глубокого погружения в процесс разработки. Он отвечает на вопросы: “Что это за релиз?”, “Можно ли проверить подлинность?”, “Из чего он состоит?”, “Как поставляются обновления?”, “Как обрабатываются уязвимости?”.

#	Что должно быть доступно	Что именно проверяем (свидетельство)	Как быстро проверить
1	Идентификация релиза	Версия продукта, дата/время сборки (если фиксируется), идентификатор сборки, состав артефактов релиза.	На странице релиза: версия + release notes + список артефактов (ISO/RAW/репозиторий/контейнеры). В системе: `cat /etc/os-release` (или эквивалентный идентификатор релиза).
2	Манифест артефактов	Единый документ с перечнем артефактов, их хэшами, ссылками на подписи и публичные ключи проверки.	Скачать `release-manifest` (по политике) и убедиться, что там есть: SHA256/SHA512 + ссылки на подписи/ключи.
3	Публичный ключ проверки	Публичный ключ (или ключи) подписи релиза с отпечатком (fingerprint) и инструкцией проверки.	Сверить fingerprint на странице доверия/релиза. Проверить ключ: `gpg --show-keys` (или RPM-ключ).
4	Проверка подписи образа	Подлинность ISO/RAW (или иных образов) через проверяемую подпись и контрольные суммы.	Проверить хэш: `sha256sum -c` (по файлу сумм). Проверить подпись: `gpg --verify` (по файлу подписи).
5	Проверка подписи пакетов/репозитория	Наличие подписи RPM и/или подписанных репозиторных метаданных; наличие доверенного ключа в системе.	Проверка RPM: `rpmkeys --checksig <pkg.rpm>`. Проверка доверенного ключа: `rpm -qa gpg-pubkey*` (или политика репозитория).

#	Что должно быть доступно	Что именно проверяем (свидетельство)	Как быстро проверить
6	Осмысленная политика обновлений	Описание каналов (например: stable/LTS/testing) и правила попадания изменений в stable (gates, approvals).	На странице политики обновлений: каналы + критерии продвижения + сроки поддержки веток.
7	SBOM релиза	Доступный SBOM, привязанный к релизу, с возможностью проверки целостности/подлинности SBOM.	Скачать SBOM, сверить идентификатор релиза, проверить подпись/хэш SBOM как артефакта релиза.
8	Security advisory	Публичный список advisory: затронутые версии, влияние, mitigation, “Fixed in”, инструкции обновления, ссылки на пакеты.	Открыть страницу advisory и убедиться, что есть минимум: affected/fixed, CVSS (если применимо), инструкции обновления, подпись/хэши.

Типовые “красные флаги”
Проверка должна считаться неуспешной при наличии хотя бы одного признака: (1) отсутствуют подписи или невозможно получить публичный ключ проверки; (2) отсутствует манифест с хэшами артефактов; (3) SBOM не привязан к релизу или не верифицируется; (4) отсутствует политика обновлений и/или отсутствует список advisory; (5) нет ясного указания поддерживаемых веток и сроков поддержки.

10.2 Чек-лист для заказчика КИИ (артефакты БРПО)

Чек-лист предназначен для проверки наличия подтверждающих материалов по безопасной разработке, испытаниям и поддержке безопасности. Он ориентирован на практику применения в строгих контурах: наличие документации, отчётов и протоколов важнее “слов о безопасности”.

Артефакт	Что должно содержать	Как проверяется
Руководство/политика БРПО (SSDLC)	Этапы процесса, роли и ответственность, критерии допуска релиза (security gates), правила критических изменений, правила управления уязвимостями, правила поставки обновлений и проверки подписи.	Наличие документа; актуальность по версии процесса; наличие ссылок на реальные артефакты и процедуры.

Артефакт	Что должно содержать	Как проверяется
Security Requirements (SRD) + критерии приёмки	Проверяемые требования безопасности и критерии допуска релиза; привязка к компонентам/подсистемам; правила эскалации.	Просмотр SRD; наличие трассируемости требований к проверкам и релизным решениям.
Анализ угроз / модель угроз	Активы, нарушители, сценарии атак, границы доверия, поверхность атаки, риски и меры снижения.	Наличие документа; наличие обновлений при существенных изменениях; наличие связи с планом испытаний.
Архитектурное описание подсистем и интерфейсов	Подсистемы, интерфейсы, порты/протоколы, потоки данных, точки расширения, привилегированные операции, пути обновления.	Наличие документа; наличие инвентаризации интерфейсов как объекта испытаний.
Отчёты SAST	Конфигурация/профиль, версия правил, перечень выявлений, решения (исправлено/ложноположит./принято), итоги на релизной ветке.	Наличие отчётов по релизу; соответствие критериям допуска; наличие связки с тикетами/фиксами.
Отчёты fuzzing	Цели фаззинга (интерфейсы/парсеры), конфигурация, длительность/покрытие (по политике), найденные краши и их устранение, регресс-наборы по найденным дефектам.	Наличие отчётов; наличие подтверждений устранения; наличие регресс-кейсов.
Отчёты DAST (если требуется профилем)	Описание стенда, цели сканирования/тестирования, результаты, подтверждение устранения критичных находок.	Наличие отчётов; воспроизводимость стенда; связь находок с исправлениями.
Процессы поддержки безопасности (VDP/PSIRT)	Каналы приёма, triage, SLA по критичности, выпуск advisory, поставка обновлений, backport в LTS, проверка подписи обновлений.	Наличие политики; наличие истории advisory/обновлений; выборочная проверка “тикет фикс релиз”.
Релизный протокол + манифест + подписи	Freeze, результаты обязательных проверок, решение об утверждении, подпись, публикация; перечень артефактов и хэшей.	Наличие релизного чек-листа; наличие манифеста; проверяемость подписи и ключей.

Артефакт	Что должно содержать	Как проверяется
SBOM релиза	Полный состав релиза, привязка к версии, возможность сопоставить SBOM пакет релиз.	Наличие SBOM; проверка идентификатора релиза; проверка целостности/подлинности SBOM.

10.3 Матрица трассируемости (требование доказательство)

Матрица трассируемости используется для аудита: каждое требование должно иметь как минимум один подтверждающий артефакт и метод проверки. Ниже приведён рекомендованный формат. Идентификаторы требований должны быть единообразными (например: `SR-xxx`, `SC-xxx`).

ID	Источник	Требование (кратко)	Доказательство (документ/лог/артефакт)	Метод проверки	Статус
SR-001	Норматив / политика	Обязательный SAST на релизной ветке	Отчёт SAST релиза; конфигурация правил; ссылки на тикеты по критичным находкам	Проверка отчёта; сопоставление с критериями допуска; выборочная проверка исправлений	Implemented
SR-002	Норматив / политика	Обязательный fuzzing для критичных интерфейсов	Отчёт fuzzing; перечень целей/интерфейсов; подтверждение устранения крашей; регресс-кейсы	Проверка отчёта; проверка наличия регресса; выборочная пересборка/перепрогон	Implemented
SC-010	Supply chain	Запрет сетевых скачиваний в build-time	Политика сборки; конфигурация билд-агентов; отчёт “no network in build”; логи блокировок	Проверка конфигураций; выборочная сборка под контролем сетевых правил	Implemented
SC-020	Supply chain	Подпись артефактов релиза и публикация ключей проверки	Release manifest с хэшами; подписи; публичный ключ с fingerprint; инструкция проверки	Независимая проверка подписи и хэшей на загруженных артефактах	Implemented
VM-001	VDP / support	Публикация advisory и поставка исправлений через каналы обновлений	Страница advisory; тикеты уязвимостей; релизные заметки security-обновлений; подписанные пакеты	Сопоставление advisory пакеты “Fixed in”; проверка подписи обновлений	Implemented

Практика применения матрицы

Для аудита достаточно выбрать 3–5 требований из разных областей (разработка, испытания, supply chain, релиз, уязвимости) и пройти полный путь доказательств до конкретного релиза: `ID → артефакт → проверка → результат`. Это даёт проверку “на реальных данных”, а не по декларациям.

11. Практические примеры

CASES

Раздел демонстрирует работу процесса безопасной разработки НАЙС.ОС на типовых сценариях. Примеры приведены в формате `событие → последовательность действий → подтверждающие материалы`. Цель — показать, что процесс “работает на земле”: есть triage, фиксы, испытания, релиз и последующее уведомление потребителей.

Примечание
В примерах используются условные обозначения (“библиотека *X*”, “уязвимость *CVE-YYYY-NNNN*”). Конкретные названия и идентификаторы зависят от реальных событий релизного цикла.

11.1 Мини-кейс: уязвимость в библиотеке X → действия НАЙС.ОС

Сценарий: опубликована уязвимость в библиотеке X, которая входит в состав поставки НАЙС.ОС (неважно, как именно пришла информация: upstream advisory, CVE, сообщение исследователя, мониторинг уязвимостей). Требуется выпустить исправление и довести информацию до пользователей.

#	Этап	Что делает НАЙС.ОС	Чем подтверждаем
1	Регистрация события	Создаётся карточка уязвимости (тикет) с привязкой к компоненту X и источнику информации (CVE/upstream/advisory/исследователь). Определяется ответственный (PSIRT) и начальный статус.	Тикет `VULN-*`; ссылка на источник (CVE/advisory); метки компонента/веток.
2	Triage и воспроизведение	Проверяется применимость: присутствует ли версия X в релизах/ветках; воспроизводится ли дефект на стенде; определяются условия эксплуатации (конфигурации, включённые функции, внешние интерфейсы).	Протокол воспроизведения; список затронутых версий; “affected / not affected” по веткам.

#	Этап	Что делает НАЙС.ОС	Чем подтверждаем
3	Оценка критичности	Рассчитывается CVSS-вектор и балл; определяется критичность и приоритет. Если доступен CWE/класс дефекта — фиксируется для анализа и отчётности.	CVSS-вектор/балл; CWE (если применимо); решение по приоритету и срокам (SLA).
4	Стратегия исправления	Выбирается стратегия: (a) обновление до исправленной версии upstream, (b) backport патча в текущую версию, (c) временная мера снижения риска (mitigation) до выхода патча. Для LTS определяются ветки backport.	Решение в тикете; план работ; список веток и пакетов; ссылка на upstream patch/релиз.
5	Реализация фикса	Мейнтейнер пакета обновляет spec/исходники, добавляет/обновляет патчи, фиксирует хэши источников, обновляет changelog и привязку к тикету/CVE. Проводится обязательный review.	Merge request; diff spec/патчей; ссылка на тикет/CVE; результат review (approve).
6	Тесты и security-верификация	Выполняются: unit/integration (если применимо), регрессионные тесты, проверка сборки и зависимостей, а также контрольные проверки безопасности по профилю компонента (SAST/фаззинг/DAST — где применимо). Дополнительно выполняется проверка, что уязвимость закрыта (re-test).	Отчёты CI; отчёты SAST/фаззинга (если применимо); протокол re-test; результаты сборки.
7	Выпуск обновления	Релиз-инженер формирует обновление для нужных веток (stable/LTS), выполняет релизный чек-лист, подписывает пакеты/репозиторные метаданные, публикует обновление в официальном канале.	Релизный протокол/чек-лист; подписи; записи публикации; идентификаторы сборок пакетов.
8	Advisory и уведомление	Публикуется advisory: описание, затронутые версии, влияние, mitigation (если требуется), “Fixed in”, инструкции обновления, способы проверки подписи/целостности обновлений.	Опубликованный advisory; ссылки на пакеты/репозиторий; CVE (если присвоен); даты и статусы.
9	Закрытие и пост-аналитика	Тикет закрывается после подтверждения публикации и успешного обновления на контрольных стендах. При необходимости добавляется регрессионный тест и обновляются правила проверок (например, SAST).	Статус “closed”; отчёт/заметка postmortem (если требуется); ссылка на регресс-тест.

Контрольный вопрос аудитора
Для выборочной проверки достаточно взять один security-advisory и проследить цепочку: advisory пакеты/версии подпись тикет MR отчёты CI протокол re-test.
Отсутствие любого элемента означает разрыв доказательной базы.

11.2 Мини-кейс: изменение сетевых дефолтов threat model и тесты

Сценарий: предлагается изменить сетевые значения по умолчанию (например, правила firewall, открытые порты, включение/выключение форвардинга, режимы сервисов). Такое изменение влияет на поверхность атаки и поэтому проходит как критическое изменение с обязательным анализом угроз и расширенной верификацией.

#	Этап	Что делаем	Чем подтверждаем
1	Классификация изменения	Изменение маркируется как критическое (влияет на surface/границы доверия). Фиксируется цель изменения (совместимость, требование заказчика, устранение риска, изменение модели эксплуатации).	Тикет/предложение; метка “critical change”; обоснование и критерии успеха.
2	Актуализация threat model	Обновляются: поверхность атаки (порты/протоколы/интерфейсы), границы доверия и сценарии нарушителя. Выполняется оценка: что становится проще атаковать, какие компенсирующие меры нужны.	Изменения в документе “Threat Model”; diff поверхности атаки; решение Security Owner.
3	Проектное решение	Выбирается вариант дефолта и его безопасные ограничения: минимальные открытые сервисы, явное включение опасных режимов, документирование необходимых админ-действий для расширения сетевой доступности.	“Security Design Decision”; обновление документации по дефолтам; миграционные заметки (если breaking change).

#	Этап	Что делаем	Чем подтверждаем
4	Реализация и review	Вносятся изменения в системные политики/конфиги/пакеты. Обязателен независимый review, участие Security Owner для утверждения изменения дефолтов.	MR/коммит; результат review; отметка Security Owner “approved”.
5	Тесты функциональные	Проверяются сценарии: чистая установка, базовая доступность (например, SSH), отсутствие лишних слушателей, корректная работа заявленных сервисов, совместимость с документацией.	Отчёты CI; протокол тестирования установки; результаты проверок открытых портов/сервисов.
6	Security-верификация конфигурации	Прогон проверок конфигураций по базовым бенчмаркам (CIS-подобная логика): firewall, порты, политики, параметры протоколов. При необходимости — DAST для сервисов, затронутых изменением.	Отчёты конфигурационных проверок; отчёты DAST (если применимо); решение по отклонениям.
7	Релиз и коммуникация	Изменение включается в релиз с явным описанием: что изменилось, как влияет на эксплуатацию, какие действия нужны для отклонения от дефолта. Обновляются release notes и документация.	Release notes; релизный чек-лист; обновлённая документация; подтверждение публикации.

Критерий качества примера

Для аудитора критично увидеть не “правильные слова”, а связку: `обоснование → обновление threat model → проектное решение → тесты → релизные материалы`. Это показывает, что изменения дефолтов проходят через управляемый процесс и подтверждаются доказательствами.