Методика обеспечения информационной безопасности программного обеспечения на основе DevSecOps: интеграция автоматизированных инструментов в жизненный цикл разработки

Н.В. Гульмамедов

Петербургский государственный университет путей сообщения Императора Александра I

Аннотация: Статья посвящена актуальным проблемам информационной безопасности при разработке программного обеспечения (ПО). Основной целью исследования является повышение уровня безопасности при разработке ПО путем внедрения инструментов безопасности и практики DevSecOps. Рассматриваются основные угрозы информационной безопасности, возникающие на каждом этапе разработки ПО начиная от планирования и заканчивая эксплуатацией продукта. Проводится анализ основных инструментов, используемых для обеспечения общей безопасности разрабатываемого ПО. Предложенная методика позволяет встроить меры безопасности в процесс разработки, минимизировать человеческий фактор, сократить время реакции на уязвимости и обеспечить контроль информационной безопасности на протяжении всего жизненного цикла ПО.
Ключевые слова: безопасная разработка, жизненный цикл, угрозы, инструменты безопасной разработки, информационная безопасность.

Введение

В различных компаниях на текущий момент значительно увеличивается потребность автоматизации ручных процессов. Это связано с увеличением требований к рабочим процессам, а также ускорения выполнения рутинных задач. Для решения этой проблемы ведутся разработки программного обеспечения, решающего ряд необходимых задач.

Разработка программного обеспечения является сложным многоступенчатым процессом, который включает анализ требования, написание технического задания, проектирование архитектуры, написание кода, тестирования и последующее внедрение с оказанием поддержки [1].

Анализ угроз информационной безопасности. В процессе разработки программного обеспечения выделяют огромное количество угроз. Проведем анализ основных угроз информационной безопасности в таблице 1.

Таблица №1.

Основные угрозы информационной безопасности

Угроза	Описание
Не полное техническое задание	Разработка некачественного технического задания к ПО, не затрагивающего все аспекты и не отвечающая требованиям политик
Несоблюдение требований безопасности	Раскрытие информации о продукте и об архитектуре ПО
Раскрытие информации о ПО	Раскрытие важной информации о программном обеспечении, а именно данные об его основных компонентах и интеллектуальной собственности, что может привести к компрометации.
Использование сторонних компонентов и инструментов разработки из ненадёжных или непроверенных источников	Появление уязвимостей в программном обеспечении
Разглашение информации в ходе тестирования на безопасность	Раскрытие слабых мест и уязвимостей в ПО злоумышленникам
Обнаружение новых уязвимостей при обновлении версии продукта	Увеличение поверхности атаки, снижение уровня безопасности обновлённого ПО

Для снижения уровня рассмотренных угроз применим методику и инструменты безопасной разработки, также известных как DevSecOps (development (разработка), security (безопасность) и operations (эксплуатация или операции)) [2, 3]. Эта методика является набором процессов и процедур, охватывающих весь жизненный цикл разработки нового программного продукта [4, 5]. В таблице 2 описаны этапы жизненного цикла программного обеспечения с применением методики DevSecOps.

Таблица №2.

Этапы жизненного цикла ПО

Этап	Содержание этапа
Планирование	• Разработка плана; • Анализ требований; • Моделирование угроз; • Формирование политик безопасности.
Разработка	• Разработка требований к ПО; • Проектирование архитектуры; • Написание кода в соответствии с требованиями.
Юнит-тестирование	• Тестирование отдельных модулей на соответствие требованиям.
Сборка	• Анализ структуры ПО; • Статическое тестирование безопасности (SAST).
Публикация	• Утверждение рабочего процесса; • Динамическое тестирование безопасности (DAST).
Установка и обновление	• Установка ПО у пользователя; • Поддержка и выпуск обновлений до новых версий.
Эксплуатация продукта	• Использование ПО в рабочей среде конечными пользователями.
Мониторинг и обратная связь	• Наблюдение за работой системы; • Сбор данных об инцидентах, уязвимостях и отзывах пользователей.

Анализ инструментов. Проанализируем ряд автоматизированных систем, которые могут быть применены на этапах жизненного цикла ПО для нахождения уязвимостей и поддержания уровня защищенности, описанных в таблице 2.

В таблице 3 представлены инструменты DevSecOps, которые помогают создавать и поддерживать системы.

Таблица №3.

Инструменты DevSecOps

Категория	Инструменты
Для статического анализа кода	SonarQube, Semgrep, Checkstyle
Для динамического тестирования	Aikido Security, Intruder, Acunetix, Checkmarx DAST
Для моделирования угроз	Irius Risk, Pirani, GRC Toolbox
Для выполнения анализа на этапе сборки	OWASP Dependency-Check SourceClear, Retire.js
Для сканирования образов Docker	Clair, Anchore, Trivy
Для обеспечения безопасности сред развертывания	Osquery, Falco

Рассмотрим инструменты, предложенные автором для внедрения в процесс разработки ПО. Начнем с системы Jenkins, объединяющей другие инструменты.

Jenkins – сервис с открытым исходным кодом на языке программирования Java, который обеспечивает процесс непрерывной интеграции и доставки программного обеспечения [6].

Инициирование работы конвейера (запуска сборки), может происходить под воздействием разных триггеров, таких как: фиксация кода в репозиторий, в соответствии с расписанием, по запросу или вследствие успешного завершения работы другого конвейера.

Контроль доступа проводится путем проверки подлинности пользователя и авторизацией. Разработчики сервиса также продумали защиту и от внешних угроз, так как подобные сервисы привлекают

злоумышленников, в связи с этим он защищен от вредоносных сборок разрабатываемого ПО и от CSRF атак (Cross-Site Request Forgery, межсайтовая подделка запроса).

Данный проект с открытым исходным кодом считается образцовым и на сегодняшний день он остаётся одним из самых популярных и мощных инструментов реализации непрерывной интеграции и доставки. Потенциал данного инструмента расширяем с помощью плагинов.

В Jenkins есть возможность интегрировать различные системы, например такие как статический анализатор кода SonarQube [6, 7] и инструмент динамического тестирования безопасности приложений OWASP ZAP [8].

Анализатор способен проводить и проверки на безопасность кодовой базы. В предполагаемом сценарии методики, проблемы качества и безопасности могут быть выявлены на раннем этапе разработки ПО. Это происходит, как только исходный код был написан разработчиком, отправлен в систему контроля версий и запущена сборка в Jenkins.

Разработчик, сосредотачивается на поддержании высоких стандартов и берет на себя ответственность конкретно за новый код, над которым работает. SonarQube предоставляет инструменты для установления высоких стандартов за то, что код соответствует этим стандартам.

Статический анализ выполняется на этапе написания кода. Это позволяет разработчикам находить недостатки на ранних этапах разработки продукта и снижать затраты на их устранение. Преимущества SAST заключаются в его возможности работы с большинством языков программирования, интегрирования в процесс разработки, обнаружение критических уязвимостей и указание точного местоположения подозрительного фрагмента кода. Последний пункт чрезвычайно важен для проектов, которые состоят из большого количества строк кода [9].

Динамическое тестирование безопасности ПО, известное как DAST, используется для воспроизведения действий атак, направленных на выявление распространенных уязвимостей [10]. Главная цель DAST заключается в выявлении недостатков и уязвимостей приложений до того, как ими воспользуются злоумышленники.

IriusRisk — полноценная платформа моделирования угроз. Поддерживает автоматическое создание угроз в реальном времени и соответствует международным стандартам, таким как OWASP.

Trivy — универсальный сканер уязвимостей Docker образов. Этот сканер проверяет слои образа (включая базовый слой) на наличие известных уязвимостей и ошибок.

Osquery — проект, разработанный с целью мониторинга и аналитики проходящих процессов в ОС. Записывает всю информацию, полученную из ОС в таблицы, и предлагает инструменты их анализа на основе языка запросов SQL.

Результаты. Таким образом, на основе проведенного анализа инструментов предлагается следующая методика обеспечения безопасности при разработке программного обеспечения. Основная суть методики заключается во встраивании автоматизированных инструментов безопасности в каждый этап жизненного цикла разработки программного обеспечения.

Цель методики – выявлять уязвимости как можно раньше и поддерживать высокий уровень защищенности на всех стадиях.

В таблице 5 представлены этапы жизненного цикла и соответствующие инструменты для обеспечения информационной безопасности.

Таблица №5.

Методика обеспечения безопасности.

Этап жизненного цикла ПО	Инструменты и меры безопасности	Цель применения
Планирование	1. Моделирование угроз с использованием IriusRisk; 2. Формирование политик безопасности	Выявление потенциальных рисков и определение требований к защите на ранней стадии.
Разработка	1. Интеграция SonarQube в IDE и CI/CD; 2. Применение подхода «Clean as You Code»	Обеспечение качества и безопасности кода в процессе написания; выявление уязвимостей сразу после их появления
Юнит-тестирование	Автоматизированные unit-тесты с проверкой безопасного поведения компонентов	Подтверждение корректности и устойчивости отдельных модулей к атакам
Сборка	1. Jenkins как центральный оркестратор CI/CD; 2. SonarQube для статического анализа (SAST)	Контроль качества сборки, блокировка выпуска при несоответствии метрикам безопасности (рейтинг безопасности ниже А (табл.4))
Публикация	1. OWASP ZAP (DAST) для динамического тестирования; 2. Интеграция ZAP в пайплайн Jenkins	Выявление уязвимостей в работающем приложении до релиза
Установка и обновление	1. Сканирование Docker-образов с помощью Trivy; 2. Проверка целостности обновлений	Предотвращение внедрения уязвимых или скомпрометированных компонентов
Эксплуатация продукта	Мониторинг среды выполнения с помощью Osquery	Обнаружение подозрительной активности и аномалий в реальном времени
Мониторинг и обратная связь	1. Сбор инцидентов и комментариев от пользователей; 2. Автоматическое обновление баз знаний об уязвимостях	Непрерывное улучшение безопасности на основе реальных данных

Методика DevSecOps значительно повышает безопасность в процессе разработки, но у данного подхода, есть ряд недостатков. Одним из которых является зависимость от большого количества внешних сервисов, это может стать проблемой для поддержания стабильности выпуска. Также ограничение

по возможностям статических и динамических анализаторов, зачастую они поддерживают ограниченный (хоть и довольно большой) список технологий для анализа. Тестирование на безопасность с помощью разных инструментов доступно даже в автоматизированном режиме, но существует риск ложных срабатываний и обычно такие инструменты поддерживают конкретную технологию.

Для совершенствования методики DevSecOps с учетом выявленных недостатков, предлагается в процессе разработки заменить ряд инструментов, на использование агента искусственного интеллекта. Современные большие языковые модели, поддерживают огромное количество языков программирования, что позволит обрести независимость перед технологическим стеком продукта. Модели могут дополнительно обучаться с помощью различных подходов. Одним из таких подходов является расширение контекста с помощью внешних источников, например, сбор информации из баз данных уязвимостей CVE или ФСТЭК для повышения качества анализа. Модели, обученные на большом количестве данных, могут существенно снизить риск ложноположительных сигналов или пропущенных уязвимостей.

Агент должен быть встроен в процесс разработки таким образом, чтобы сканирование кода и конфигураций происходило до его сборки, а активное тестирование уже после получения артефактов для запуска. Алгоритм работы разрабатываемого агента состоящего из этапов сканирования и тестирования, представлен ниже.

Сканирование на безопасность

– Сбор и подготовка данных: Сбор необходимой информации: конфигурации, исходный код, документацию, комментарии, предыдущие анализы, а также информацию о проекте и его структуре.

– Расчет метрик (показателей) для исходного кода. Метрики предлагается использовать для улучшения качества анализа исходного кода.

– Внедрение результатов расчета метрик в инструкцию для модели и отправка файлов на анализ. Каждый файл исходного кода отправляется отдельно на анализ в соответствии с инструкцией.

– Языковая модель выдаёт рекомендации по улучшению кода (валидный список объектов JSON), оптимизацию конфигурации и меры повышения безопасности. Данные сохраняются в базу данных, для использования в будущих обработках.

– Система должна будет по итогам работы принять решение на основании полученных выводов, о продолжении сборки, проведении дополнительных проверок или возврате к доработкам.

Тестирование на безопасность

– Сбор необходимой информации: конфигурации, исходный код, документацию, комментарии, предыдущие тесты, а также информацию о проекте и его структуре.

– Генерация плана тестирования на основе контекста. Важно определить класс ПО, язык и способы тестирования. Предлагается дать возможность модели использовать различные варианты, и принимать решение об использовании конкретного способа самостоятельно.

– По плану тестирования модель генерирует команды, файлы или другую информацию для проведения тестирования. После каждого выполнения команды, модель получает результаты ее выполнения. В случае ошибок, модель должна исправить команду и попробовать снова, пока не будет выполнена цель плана. Основным преимуществом данного подхода, является возможность проверки найденных уязвимостей на возможность эксплуатации и полная автоматизация.

– Языковая модель выдаёт рекомендации по исправлению проблем, оптимизацию конфигурации и меры повышения безопасности.

– Система должна будет по итогам работы принять решение на основании полученных выводов, о проведении дополнительных тестов, возврате к доработкам или дальнейшей публикации в производственной среде.

Заключение. Ключевые принципы методики можно сформулировать следующим образом:

1. Раннее выявление уязвимостей, то есть безопасность проверяется начиная с 1-го этапа.
2. Минимизация ручного труда и человеческого фактора за счёт CI/CD-интеграции.
3. Защита обеспечивается на всех этапах.
4. Блокировка небезопасных сборок осуществляется за счет соблюдения пороговых метрик (см. таблицу 4).

Таким образом, предложенная методика позволяет встроить безопасность в процесс разработки, минимизировать человеческий фактор, сократить время реакции на уязвимости и обеспечить контроль информационной безопасности на протяжении всего жизненного цикла ПО.

Литература

1. Управление разработкой программного обеспечения: организация процесса, модели, инструменты. URL: simpleone.ru/blog/how-to-organize-the-software-development-process (дата обращения: 17.01.2026).

2. Бархатов А. Что такое DevSecOps: практики, методология, инструменты и почему бизнесу стоит его внедрять в разработку? URL: timeweb.cloud/blog/devsecops-praktiki-metodologiya-instrumenty (дата обращения: 17.01.2026).

3. Виноградов М. А. Интеграция безопасности в DevOps (DevSecOps) и автоматизация безопасности на протяжении всего жизненного цикла разработки программного обеспечения // СЕВЕРГЕОЭКОТЕХ-2025: Материалы XXVI Международной молодёжной научной конференции. Ухта, 2025. С. 276-281.

4. Гульмамедов Н. В., Корниенко А. А., Барабанщикова М. Н. Повышение эффективности защиты информации при разработке программного обеспечения с использованием методологии DevSecOps // Цифровые системы и модели: теория и практика проектирования, разработки и использования: Материалы международной научно-практической конференции. Казань: Казанский государственный энергетический университет, 2025. С. 2113-2117.

5. Sinan M., Shahin M., Gondal I. Integrating Security Controls in DevSecOps: Challenges, Solutions, and Future Research Directions // Journal of Software: Evolution and Process. 2025. V. 37. №6. URL: doi.org/10.1002/smr.70029.

6. Pandi S. S., Kumar P., Suchindhar R. M. Integrating jenkins for efficient deployment and orchestration across multi-cloud environments // 2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES). IEEE, 2023. pp. 1-6.

7. Корниенко А. А, Корниенко С. В., Гульмамедов Н. В., Собакин С. В. Адаптация методологии DevOps с конвейером CI\CD для обеспечения безопасности автоматизированной системы управления производственной деятельностью // Двойные технологии. 2025. № 1(110). С. 75-81.

8. Putri V. R., Sobandi A., Santoso B. Analysis of Information System Security Using OWASP ZAP on a Web-Based Electronic Archiving System // Telematika. 2025. V. 22. №3. pp. 28-42.

9. Селиверстов С.Д., Мироненко Ю.В., Гниденко И.Г. Обзор методологии DevSecOps и ее ключевых инструментов для внедрения и обеспечения безопасной разработки ПО // СТУДЕНТ ГОДА 2024: Сборник статей

Международного научно-исследовательского конкурса. Пенза, 2024. С. 107-111.

10. Комисарчук А., Трапезников Е.В. Тестирование безопасности приложений на ранних этапах разработки // Нанотехнологии. Информация. Радиотехника (НИР-22). 2022. С. 69-74.

References

1. Upravlenie razrabotkoj programmnogo obespecheniya: organizaciya processa, modeli, instrumenty [Software development management: process organization, models, tools]. URL: simpleone.ru/blog/how-to-organize-the-software-development-process (accessed: 17.01.2026).

2. Barhatov A. Chto takoe DevSecOps: praktiki, metodologiya, instrumenty i pochemu biznesu stoit ego vnedryat' v razrabotku? [What is DevSecOps: practices, methodology, tools, and why should a business implement it in development?]. URL: timeweb.cloud/blog/devsecops-praktiki-metodologiya-instrumenty (accesseed: 17.01.2026).

3. Vinogradov M. A. Integraciya bezopasnosti v DevOps (DevSecOps) i avtomatizaciya bezopasnosti na protyazhenii vsego zhiznennogo cikla razrabotki programmnogo obespecheniya [Security integration in DevOps (DevSecOps) and automation of security throughout the software development lifecycle] (SEVERGEOEKOTEH-2025: Materialy XXVI Mezhdunarodnoj molodyozhnoj nauchnoj konferencii). Uhta, 2025. pp. 276-281.

4. Gul'mamedov N. V., Kornienko A. A., Barabanshchikova M. N. Povyshenie effektivnosti zashchity informacii pri razrabotke programmnogo obespecheniya s ispol'zovaniem metodologii DevSecOps [Improving the effectiveness of information protection in software development using the DevSecOps methodology] (Cifrovye sistemy i modeli: teoriya i praktika proektirovaniya, razrabotki i ispol'zovaniya : Materialy mezhdunarodnoj nauchno-prakticheskoj

konferencii). Kazan' : Kazanskij gosudarstvennyj energeticheskij universitet, 2025. pp. 2113-2117.

5. Sinan M., Shahin M., Gondal I. Journal of Software: Evolution and Process. 2025. V. 37. №6. URL: doi.org/10.1002/smr.70029.

6. Pandi S. S., Kumar P., Suchindhar R. M. Integrating jenkins for efficient deployment and orchestration across multi-cloud environments (2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)). IEEE, 2023. pp. 1-6.

7. Kornienko A. A., Kornienko S. V., Gul'mamedov N. V., Sobakin S. V. Dvojnye tekhnologii. 2025. № 1(110). pp. 75-81.

8. Putri V. R., Sobandi A., Santoso B. Telematika. 2025. V. 22. №3. pp. 28-42.

9. Seliverstov S.D., Mironenko Yu.V., Gnidenko I.G. Obzor metodologii DevSecOps i ee klyuchevyh instrumentov dlya vnedreniya i obespecheniya bezopasnoj razrabotki PO [Review of the DevSecOps methodology and its key tools for implementing and ensuring secure software development] (STUDENT GODA 2024: Sbornik statej Mezhdunarodnogo nauchno-issledovatel'skogo konkursa). Penza, 2024. pp. 107-111.

10. Komisarchuk A., Trapeznikov E.V. Nanotekhnologii. Informaciya. Radiotekhnika (NIR-22). 2022. pp. 69-74.

Авторы согласны на обработку и хранение персональных данных.

Дата поступления: 5.01.2026

Дата публикации: 3.03.2026