

РАЗРАБОТКА ЗАЩИЩЕННОГО КОНВЕЙЕРА CI/CD

З.А. Богданчиков¹, А.С. Ершов¹

¹*Саратовский государственный технический университет им. Гагарина Ю.А.,
г. Саратов
e-mail: sacharb@ya.ru*

Аннотация

В работе рассматриваются подходы к интеграции инструментов информационной безопасности в CI/CD-конвейеры в контексте концепции DevSecOps. Проведён обзор средств статического и динамического анализа кода, анализа зависимостей и поиска секретов, а также рассмотрены особенности их встраивания в автоматизированные процессы разработки. Показано, что применение данных инструментов реализует принцип shift-left, позволяя выявлять уязвимости на ранних этапах и тем самым снижать риск их попадания в производственную среду.

Ключевые слова: CI/CD, DevSecOps, безопасность программного обеспечения, конвейер разработки, интеграция инструментов безопасности, статический анализ, динамический анализ, принцип shift-left.

Введение

В настоящее время процессы разработки программного обеспечения характеризуются высокой скоростью выпуска новых версий и активным использованием автоматизации. Широкое распространение получили подходы Continuous Integration и Continuous Delivery (CI/CD), позволяющие значительно сократить время разработки и повысить качество программных продуктов. Однако ускорение жизненного цикла разработки нередко приводит к снижению уровня контроля информационной безопасности, что увеличивает риск появления уязвимостей в программном обеспечении.

Современные угрозы информационной безопасности, такие как утечки данных, внедрение вредоносного кода и эксплуатация уязвимостей сторонних библиотек, требуют пересмотра традиционных подходов к обеспечению безопасности. В условиях CI/CD применение исключительно ручных проверок становится неэффективным, что обуславливает необходимость внедрения автоматизированных средств безопасности непосредственно в процессы разработки. В связи с этим все большую актуальность приобретает концепция DevSecOps, предполагающая интеграцию механизмов информационной безопасности на всех этапах жизненного цикла программного обеспечения [1].

Интеграция инструментов безопасности в CI/CD процессы позволяет реализовать принцип «shift-left», при котором выявление уязвимостей осуществляется на ранних этапах разработки, что снижает затраты на их устранение и повышает общий уровень защищенности программных продуктов [2].

В данной работе предлагается пример реализации CI/CD конвейера, содержащей инструменты статического и динамического анализа кода, проверки зависимостей и поиска секретов.

CI/CD конвейер и DevSecOps

CI/CD конвейер представляет собой последовательность автоматизированных этапов, выполняемых при внесении изменений в исходный код. Конвейер описывает полный путь программного продукта от момента фиксации изменений в системе контроля версий до развертывания в целевой среде.

Каждый этап конвейера состоит из набора задач, выполняемых в изолированных средах. В современных системах непрерывной интеграции такие задачи, как правило, запускаются в контейнерах, что обеспечивает воспроизводимость среды выполнения и упрощает масштабирование процессов сборки и тестирования. Благодаря этому в конвейер могут быть интегрированы различные скрипты, инструменты и программные компоненты, автоматизирующие отдельные этапы жизненного цикла разработки.

Интеграция инструментов обеспечения безопасности в конвейер представляет собой практическую реализацию концепции DevSecOps и принципа shift-left.

Инструменты безопасности в CI/CD

В рамках CI/CD процессов обеспечение безопасности достигается посредством применения специализированных инструментов, каждый из которых ориентирован на решение определённого класса задач и используется на соответствующем этапе конвейера.

Инструменты статического анализа защищённости кода (SAST – Static Application Security Testing). Данные инструменты осуществляют анализ исходного кода приложения без его исполнения и обеспечивают выявление потенциальных уязвимостей, в частности небезопасной обработки пользовательского ввода, применения уязвимых программных конструкций и нарушений требований безопасной разработки. Ключевым достоинством SAST является возможность обнаружения проблем безопасности непосредственно после внесения изменений в кодовую базу, то есть на наиболее ранних стадиях разработки.

Инструменты анализа состава программного обеспечения (SCA – Software Composition Analysis), предназначенные для выявления уязвимостей в сторонних библиотеках и зависимостях, задействованных в проекте. В условиях широкого распространения компонентов с открытым исходным кодом наличие уязвимости даже в одной из используемых библиотек способно представлять существенную угрозу безопасности всего приложения. Инструменты SCA в автоматическом режиме сопоставляют версии применяемых зависимостей с базами данных известных уязвимостей, что позволяет оперативно идентифицировать сопутствующие риски. [3]

Инструменты обнаружения секретов (Secrets Scanning), направленные на выявление конфиденциальных сведений – паролей, токенов доступа, ключей API – которые могут быть непреднамеренно включены в исходный код или файлы конфигурации. Применение данных инструментов позволяет предотвратить утечку учётных данных и снизить риск компрометации внешних сервисов и инфраструктурных компонентов.

Инструменты динамического тестирования защищённости приложений (DAST – Dynamic Application Security Testing) функционируют принципиально иначе: в отличие от статического анализа, DAST осуществляет проверку работающего приложения, взаимодействуя с ним по аналогии с действиями потенциального злоумышленника. Такой подход позволяет выявлять уязвимости, проявляющиеся исключительно в процессе исполнения, - в частности, межсайтовый скриптинг (XSS) и недостатки механизмов аутентификации. DAST, как правило, применяется на завершающих этапах CI/CD конвейера, после развёртывания приложения в тестовой среде.

Совокупное использование инструментов SAST, SCA, Secrets Scanning и DAST обеспечивает многоуровневую защиту программного обеспечения и позволяет охватить основные классы уязвимостей на всех стадиях CI/CD процесса.

Реализация в GitLab CI/CD

В качестве платформы для реализации CI/CD конвейера был выбран GitLab CI/CD – встроенная система автоматизации, интегрированная непосредственно в GitLab и обеспечивающая полный жизненный цикл разработки: от управления репозиторием до развёртывания приложения. Конфигурация конвейера описывается в файле `.gitlab-ci.yml`, в котором определяются стадии, задачи и используемые образы контейнеров. [4]

В реализованном конвейере содержится две стадии: сборка (build) и security (анализ безопасности). Следующая стадия в конвейере не запустится пока все задачи предыдущей стадии не выполняться успешно. Следующая стадия не начнется, если хотя бы одна задача текущей стадии завершится с ошибкой.

Статический анализ защищённости кода (SAST) выполняется на базе образа Semgrep и осуществляет сканирование исходного кода с использованием автоматически подбираемого набора правил. Автоматически сгенерированные файлы исключаются из анализа. Результаты сохраняются в формате JSON в качестве артефакта конвейера.

Анализ состава программного обеспечения (SCA) реализован в виде двух независимых задач. Сканирование зависимостей серверной части выполняется посредством инструмента Trivy: анализу подвергаются зависимости .NET-проекта с фильтрацией по уровням критичности CRITICAL и HIGH.

Проверка зависимостей клиентской части осуществляется штатной утилитой `npm audit` с аналогичным порогом критичности. Выявленные уязвимости уровня HIGH и выше также приводят к завершению задачи с ошибкой, а отчёт сохраняется в формате JSON.

Задача обнаружения секретов выполняется на базе образа Gitleaks и осуществляет сканирование репозитория на предмет наличия конфиденциальных данных – паролей, токенов доступа и ключей API. При обнаружении секретов задача завершается с ошибкой, блокируя дальнейшее выполнение конвейера. Результаты сканирования сохраняются в формате JSON в качестве артефакта конвейера.

Динамическое тестирование защищённости реализовано в виде двух независимых задач на базе официального образа OWASP ZAP. Сначала осуществляется базовое сканирование развёрнутого веб-приложения посредством `zap-baseline.py`.

Сканирование выполняется с передачей заголовка авторизации для обеспечения доступа к защищённым разделам приложения. По завершении сканирования производится анализ отчёта: при обнаружении уязвимостей уровней HIGH и CRITICAL задача завершается с ошибкой, блокируя дальнейшее выполнение конвейера. Далее реализуется сканирование REST API посредством zap-api-scan.py на основе спецификации OpenAPI. Аналогично первой задаче, передаётся заголовок авторизации, а при выявлении уязвимостей высокого и критического уровней выполнение конвейера прерывается. Результаты обеих задач сохраняются в форматах HTML и JSON в качестве артефактов конвейера.

Заключение

В представленной статье рассмотрены подходы к интеграции инструментов информационной безопасности в процессы непрерывной интеграции и доставки программного обеспечения. Были проанализированы основные принципы построения CI/CD-конвейеров и рассмотрена концепция DevSecOps, предполагающая внедрение механизмов обеспечения безопасности на всех этапах жизненного цикла разработки программного обеспечения.

В ходе работы проведен анализ этапов CI/CD-конвейера с точки зрения обеспечения безопасности, а также рассмотрены основные инструменты автоматизированного контроля, включая статический анализ исходного кода, динамический анализ приложений, анализ зависимостей и поиск секретов.

По результатам проведенного исследования был разработан пример CI/CD-конвейера, с интегрированными инструментами безопасности. Реализованный конвейер иллюстрирует практическое применение принципа shift-left, направленного на перенос процессов выявления уязвимостей на ранние этапы разработки.

В целом интеграция автоматизированных средств безопасности в CI/CD-конвейеры способствует повышению уровня защищенности разрабатываемого программного обеспечения и формированию непрерывного процесса контроля безопасности в рамках современных практик разработки.

Список использованных источников

1. Блинов, А. В. DevSecOps: объединение процессов разработки и безопасности / А. В. Блинов, С. В. Беззатеев // Вопросы кибербезопасности – 2025. – Т. 2, № 66. – С. 78–89.
2. OWASP Foundation. OWASP DevSecOps Guideline. – URL: <https://owasp.org/www-project-devsecops-guideline/> (дата обращения: 14.03.2026).
3. DevSecOps: принципы работы и сравнение SCA // Habr. – URL: https://habr.com/ru/companies/swordfish_security/articles/516660/ (дата обращения: 14.03.2026).
4. GitLab Inc. GitLab Documentation. – URL: <https://docs.gitlab.com/> (дата обращения: 14.03.2026).

DEVELOPMENT OF A SECURE CI/CD PIPELINE

Z.A. Bogdanchikov¹, A.S. Ershov¹

*¹Yuri Gagarin State Technical University of Saratov, Saratov
e-mail: sacharb@ya.ru*

Abstract

The paper examines approaches to integrating information security tools into CI/CD pipelines within the context of the DevSecOps concept. An overview of static and dynamic code analysis tools, dependency analysis, and secret detection is provided, along with an examination of the features of their integration into automated development processes. It is shown that the use of these tools implements the shift-left principle, allowing vulnerabilities to be identified at an early stage and thereby reducing the risk of them entering the production environment.

Keywords: CI/CD, DevSecOps, software security, development pipeline, integration of security tools, static analysis, dynamic analysis, shift-left principle.