Рабаданов Саид Запирович, бакалавр, 4 курс факультет Институт Информационных Технологий / кафедра Инструментального и прикладного программного обеспечения, РТУ МИРЭА, г. Москва

DEVSECOPS – SAST/DAST-ИНСТРУМЕНТЫ В CI/CD

Аннотация

В статье рассматриваются подходы к внедрению практик DevSecOps, предполагающих включение инструментов автоматизированного анализа уязвимостей в конвейеры непрерывной интеграции и доставки программного обеспечения. Основное внимание уделено применению средств статического и динамического анализа безопасности - SAST и DAST - на ранних этапах жизненного цикла разработки. В рамках исследования проведён сравнительный анализ распространённых инструментов статического анализа кода (Checkmarx, SonarQube, Semgrep) и средств динамического тестирования безопасности веб-приложений (OWASP ZAP, Burp Suite), интегрируемых в CI/CD-среды на базе Jenkins и GitHub Actions. Рассматриваются особенности их работы, архитектурные принципы интеграции и влияние на процессы автоматизированной сборки программных продуктов. Практическая часть исследования направлена на выявление эффективности данных инструментов при анализе типового сервисного приложения. В качестве ключевых метрик используются количество обнаруженных уязвимостей, уровень ложных срабатываний и временные затраты на проведение сканирования. Полученные результаты позволяют оценить влияние автоматизированного контроля безопасности на снижение числа уязвимостей, достигающих производственной среды. Практическая значимость работы заключается в формировании рекомендаций по интеграции инструментов анализа безопасности в конвейеры CI/CD и повышению уровня защищённости программных систем на ранних этапах разработки.

Ключевые слова: DevSecOps, SAST, DAST, CI/CD, безопасность программного обеспечения, автоматизированный анализ уязвимостей, Jenkins, GitHub Actions.

Annotation

This article discusses approaches to implementing DevSecOps practices, which involve integrating automated vulnerability analysis tools into software continuous integration and delivery pipelines. The main focus is on applying static and dynamic security analysis tools – SAST and DAST – at early stages of the development lifecycle. As part of the research, a comparative analysis of common static code analysis tools (Checkmarx, SonarQube, Semgrep) and dynamic web application security testing tools (OWASP ZAP, Burp Suite), integrated into CI/CD environments based on Jenkins and GitHub Actions, was conducted. The specifics of their operation, architectural principles of integration, and impact on automated software product build processes are considered. The practical part of the research aims to identify the effectiveness of these tools when analyzing a typical service application. Key metrics used include the number of detected vulnerabilities, the rate of false positives, and the time costs for scanning. The obtained results allow for an assessment of the impact of automated security control on reducing the number of vulnerabilities reaching the production environment. The practical significance of the work lies in forming recommendations for integrating security analysis tools into CI/CD pipelines and enhancing the security level of software systems at early development stages.

Keywords: DevSecOps, SAST, DAST, CI/CD, software security, automated vulnerability analysis, Jenkins, GitHub Actions.

Введение

Современные программные системы всё чаще разрабатываются в условиях высокой скорости выпуска обновлений и постоянного расширения функциональности. Использование практик непрерывной интеграции и непрерывной доставки (CI/CD) позволяет значительно ускорить процессы разработки и внедрения программного обеспечения, однако одновременно

увеличивает риск появления уязвимостей в конечных продуктах. В результате вопросы информационной безопасности становятся неотъемлемой частью жизненного цикла разработки программных систем.

Традиционные подходы к обеспечению безопасности, основанные на проведении проверок уже после завершения разработки, в современных условиях оказываются недостаточно эффективными. Обнаружение уязвимостей на поздних стадиях жизненного цикла приводит к увеличению затрат на их устранение и повышает вероятность попадания уязвимых компонентов в производственную среду. В этой связи всё более широкое распространение получает концепция DevSecOps, предполагающая интеграцию механизмов обеспечения безопасности непосредственно в процессы разработки и доставки программного обеспечения.

Одним из ключевых направлений реализации данной концепции является применение автоматизированных инструментов анализа безопасности, способных выявлять потенциальные уязвимости на ранних этапах сборки программных продуктов. Среди таких инструментов выделяются средства статического анализа исходного кода (Static Application Security Testing, SAST) и динамического тестирования безопасности приложений (Dynamic Application Security Testing, DAST). Их использование в составе конвейеров CI/CD позволяет систематически контролировать безопасность разрабатываемых систем и оперативно выявлять потенциальные угрозы.

Целью настоящего исследования является анализ эффективности применения инструментов SAST и DAST в конвейерах CI/CD и оценка их влияния на снижение количества уязвимостей, достигающих производственной среды. В рамках работы проводится сравнительное исследование ряда популярных инструментов безопасности, интегрируемых в автоматизированные процессы сборки и тестирования программных систем.

Теоретическая основа

Современные подходы к разработке программного обеспечения характеризуются высокой степенью автоматизации процессов сборки, тестирования и развертывания приложений. Использование конвейеров непрерывной интеграции и доставки (CI/CD) позволяет существенно ускорить выпуск новых версий программных продуктов за счёт автоматической обработки каждого изменения исходного кода. При этом изменения проходят последовательные этапы сборки, тестирования и проверки, что повышает эффективность разработки и сокращает время вывода программных решений на рынок.

Одновременно с ростом скорости разработки возрастает и значимость вопросов информационной безопасности. При отсутствии систематического контроля уязвимости, возникающие на этапе разработки, могут оставаться незамеченными и попадать в производственную среду. В связи с этим всё более широкое распространение получает концепция DevSecOps, предполагающая интеграцию механизмов обеспечения безопасности непосредственно в процессы разработки и эксплуатации программных систем. Данный подход основан на принципе непрерывного контроля безопасности на всех стадиях жизненного цикла программного обеспечения.

Одним из ключевых направлений реализации DevSecOps является применение автоматизированных инструментов анализа безопасности приложений. Наиболее распространёнными методами являются статический анализ исходного кода и динамическое тестирование работающих приложений. Совместное использование данных методов позволяет выявлять широкий спектр уязвимостей как на уровне программного кода, так и на уровне функционирования программной системы.

SAST (Static Application Security Testing)

Статический анализ безопасности приложений (SAST) представляет собой метод исследования программного кода без запуска самого приложения. Основная идея данного подхода заключается в анализе структуры исходного кода, логики выполнения программы и используемых программных

конструкций с целью выявления потенциальных уязвимостей ещё до этапа развертывания программного продукта. Такой подход позволяет обнаруживать проблемы безопасности на ранних стадиях жизненного цикла разработки, когда их устранение требует значительно меньших затрат ресурсов.

Принцип работы инструментов SAST основан на построении модели программы и анализе её структуры. Система анализирует исходный код, формирует абстрактное синтаксическое дерево и отслеживает возможные пути выполнения программы. На основе этого выявляются потенциально опасные участки кода, связанные с некорректной обработкой пользовательского ввода, использованием небезопасных функций, нарушениями правил работы с памятью или некорректной реализацией механизмов аутентификации и авторизации.

Одним из преимуществ статического анализа является возможность его применения на самых ранних этапах разработки. Проверка может выполняться непосредственно после внесения изменений в репозиторий исходного кода, что делает SAST эффективным инструментом для интеграции в процессы непрерывной интеграции. В рамках CI/CD-конвейеров такие проверки могут автоматически запускаться при каждом коммите или сборке проекта, обеспечивая постоянный контроль безопасности программного продукта.

Среди наиболее распространённых инструментов статического анализа безопасности можно выделить Checkmarx, SonarQube и Semgrep. Эти системы обладают широкими возможностями анализа кода, поддерживают различные языки программирования и способны выявлять множество типов уязвимостей. Помимо обнаружения проблем безопасности, многие из них также анализируют качество кода, что способствует повышению общей устойчивости программной системы.

Несмотря на высокую эффективность, статический анализ имеет и определённые ограничения. Одной из распространённых проблем является наличие ложных срабатываний, когда система сигнализирует о потенциальной

уязвимости, которая в действительности не представляет угрозы. Это требует дополнительного анализа результатов со стороны разработчиков или специалистов по информационной безопасности.

Тем не менее применение SAST остаётся важным элементом современной стратегии обеспечения безопасности программного обеспечения. Раннее выявление уязвимостей позволяет значительно снизить вероятность их попадания в производственную среду и способствует формированию более безопасного процесса разработки.

DAST (Dynamic Application Security Testing)

Динамическое тестирование безопасности приложений (DAST) представляет собой метод анализа программных систем в процессе их выполнения. В отличие от статического анализа, который исследует исходный код, динамическое тестирование ориентировано на изучение поведения функционирующего приложения. Такой подход позволяет выявлять уязвимости, проявляющиеся непосредственно во время работы системы, включая ошибки конфигурации, проблемы взаимодействия компонентов и недостатки реализации механизмов безопасности.

В рамках динамического анализа приложение рассматривается как «чёрный ящик», структура внутреннего кода которого не анализируется напрямую. Инструменты DAST взаимодействуют с системой через её интерфейсы, отправляя различные типы запросов и анализируя ответы приложения. На основе полученных данных выявляются возможные нарушения безопасности, такие как некорректная обработка входных данных, уязвимости веб-интерфейсов и ошибки реализации протоколов взаимодействия.

Одним из ключевых преимуществ динамического тестирования является возможность обнаружения уязвимостей, которые невозможно выявить при статическом анализе кода. Например, DAST эффективно выявляет проблемы конфигурации серверов, ошибки логики обработки

запросов, нарушения механизмов управления сессиями и другие проблемы, возникающие только при реальном функционировании приложения.

Среди наиболее распространённых инструментов динамического анализа безопасности можно выделить OWASP ZAP и Burp Suite. Эти системы используются для тестирования веб-приложений и позволяют автоматически сканировать интерфейсы приложения, выявляя распространённые типы уязвимостей, включая SQL-инъекции, межсайтовое выполнение сценариев (XSS), ошибки аутентификации и другие угрозы безопасности.

Инструменты DAST могут быть интегрированы в процессы непрерывного тестирования программных систем. В рамках CI/CD-конвейеров динамический анализ обычно выполняется после этапа сборки и развертывания тестовой версии приложения. Это позволяет автоматически проверять безопасность системы перед выпуском новой версии программного продукта.

Несмотря на высокую эффективность, динамическое тестирование также имеет определённые ограничения. Проведение полного анализа может требовать значительного времени, что способно замедлять процессы автоматизированной сборки. Кроме того, эффективность DAST во многом зависит от полноты сценариев тестирования и корректности настройки инструментов анализа. Однако в сочетании со статическим анализом данный подход обеспечивает более комплексную оценку уровня безопасности программного обеспечения.

Интеграция инструментов безопасности в процессы CI/CD

В современных условиях разработки программного обеспечения обеспечение безопасности становится неотъемлемой частью процессов непрерывной интеграции и доставки программных продуктов. Концепция DevSecOps предполагает внедрение механизмов автоматизированного анализа безопасности непосредственно в конвейеры CI/CD, что позволяет выполнять проверки уязвимостей на регулярной основе и выявлять потенциальные угрозы на ранних этапах разработки.

Интеграция инструментов SAST и DAST в конвейеры сборки позволяет автоматически анализировать программный код и тестировать функционирующее приложение при каждом изменении исходного кода. В результате система непрерывной интеграции выполняет не только задачи сборки и тестирования, но и функции контроля безопасности программного продукта. Такой подход значительно снижает вероятность попадания уязвимого кода в производственную среду и повышает общий уровень защищённости разрабатываемых систем.

Анализ исходного кода в процессе сборки

Одним из наиболее распространённых способов внедрения инструментов безопасности в CI/CD является автоматический запуск статического анализа исходного кода на этапе сборки проекта. После получения изменений из репозитория система непрерывной интеграции инициирует процесс компиляции и одновременно выполняет анализ кода с использованием инструментов SAST.

Инструменты статического анализа исследуют структуру программы, выявляют потенциально опасные конструкции и формируют отчёты о найденных уязвимостях. При обнаружении критических проблем система может автоматически прерывать процесс сборки, уведомляя разработчиков о необходимости устранения обнаруженных недостатков. Такой механизм позволяет предотвратить распространение уязвимого кода на последующие этапы разработки и тестирования.

Динамическое тестирование на этапе тестовой среды

После успешного завершения этапа сборки программный продукт может быть автоматически развернут в тестовой среде, где выполняется динамическое тестирование безопасности. На данном этапе применяются инструменты DAST, которые анализируют работу функционирующего приложения посредством отправки различных запросов и анализа ответов системы.

Динамический анализ позволяет выявлять уязвимости, которые невозможно обнаружить при статическом анализе исходного кода. К таким проблемам относятся ошибки конфигурации веб-сервера, нарушения механизмов аутентификации и авторизации, а также уязвимости веб-интерфейсов. Использование DAST-инструментов в рамках CI/CD обеспечивает регулярное проведение проверок безопасности перед выпуском новых версий программного обеспечения.

Использование систем автоматизации

Для организации автоматизированных процессов анализа безопасности широко применяются системы управления CI/CD, такие как Jenkins и GitHub Actions. Эти платформы позволяют создавать сценарии автоматического выполнения задач, включая сборку проекта, запуск тестов и проведение анализа безопасности.

В рамках таких систем инструменты SAST и DAST могут быть интегрированы в виде отдельных этапов конвейера. Например, после этапа компиляции может запускаться статический анализ кода, а после развертывания тестовой версии приложения - динамическое сканирование безопасности. Результаты анализа сохраняются в виде отчётов и могут использоваться для мониторинга уровня безопасности проекта.

Масштабирование процессов безопасности

В крупных программных проектах анализ безопасности может выполняться параллельно для нескольких сервисов или микросервисных компонентов. Для обеспечения масштабируемости таких процессов применяются технологии контейнеризации и оркестрации, позволяющие запускать инструменты анализа в изолированных средах.

Использование контейнеров и облачной инфраструктуры позволяет распределять задачи анализа безопасности между несколькими вычислительными узлами, сокращая время выполнения сканирования и повышая эффективность CI/CD-конвейеров. Такой подход особенно актуален

для крупных проектов, где объём исходного кода и количество сервисов могут быть значительными.

Пример архитектуры DevSecOps-конвейера

Возможная архитектура интеграции инструментов безопасности в CI/CD может включать следующие этапы:

1. Получение изменений из репозитория - разработчик отправляет изменения в систему контроля версий.
2. Сборка проекта - CI/CD-система выполняет компиляцию и подготовку артефактов.
3. Статический анализ безопасности - запуск инструментов SAST для проверки исходного кода.
4. Развертывание тестовой среды - автоматическое создание тестовой версии приложения.
5. Динамическое тестирование безопасности - сканирование приложения инструментами DAST.
6. Формирование отчётов - сохранение результатов анализа и уведомление разработчиков.
7. Развертывание в производственной среде - выпуск новой версии программного продукта при успешном прохождении всех этапов проверки.

Такой подход обеспечивает систематический контроль безопасности программных систем и позволяет выявлять потенциальные уязвимости ещё до момента развертывания приложения в производственной среде.

Сравнительный анализ инструментов SAST и DAST

В современных DevSecOps-процессах используется широкий спектр инструментов автоматизированного анализа безопасности. Каждый из них обладает собственными особенностями архитектуры, поддерживаемыми технологиями и областью применения. Для оценки эффективности применения различных инструментов целесообразно рассмотреть их основные характеристики и функциональные возможности.

Среди средств статического анализа безопасности наиболее распространены Checkmarx, SonarQube и Semgrep. Эти инструменты предназначены для анализа исходного кода и выявления потенциальных уязвимостей на ранних этапах разработки. Они поддерживают различные языки программирования и позволяют интегрировать процессы анализа непосредственно в конвейеры CI/CD.

Для динамического тестирования безопасности широко применяются OWASP ZAP и Burp Suite. Данные инструменты позволяют анализировать функционирующее приложение, выявляя уязвимости веб-интерфейсов, ошибки конфигурации и проблемы реализации механизмов аутентификации. Их применение особенно эффективно при тестировании веб-приложений и сервисов, доступных через HTTP-интерфейсы.

Организация автоматизированного запуска инструментов анализа безопасности осуществляется с использованием платформ непрерывной интеграции и доставки программного обеспечения. Среди наиболее распространённых решений можно выделить Jenkins и GitHub Actions, которые позволяют формировать конвейеры автоматической сборки, тестирования и проверки безопасности программных систем.

Ниже приведена сравнительная характеристика наиболее распространённых инструментов, используемых в рамках DevSecOps-подхода.

Критерий	Checkmarx	SonarQube	Semgrep	OWASP ZAP	Burp Suite
Тип инструмента	SAST	SAST	SAST	DAST	DAST
Основное назначение	Статический анализ безопасности кода	Анализ качества и безопасности кода	Быстрый статический анализ по шаблонам	Динамическое сканирование веб-приложений	Профессиональное тестирование веб-приложений
Принцип анализа	Анализ структуры исходного кода	Анализ кода и метрик качества	Поиск уязвимых паттернов	Тестирование работающего приложения	Интерактивное тестирование
Поддерживаемые языки	Java, C#, Python, JS	Большинство языков	Более 20 языков	Не зависит от языка	Не зависит от языка

Критерий	Checkmarx	SonarQube	Semgrep	OWASP ZAP	Burp Suite
Интеграция с CI/CD	Да	Да	Да	Да	Да
Тип уязвимостей	Ошибки кода, небезопасные конструкции	Уязвимости и проблемы качества	Типовые ошибки безопасности	XSS, SQL-инъекции	Широкий спектр веб-уязвимостей
Скорость анализа	Средняя	Средняя	Высокая	Средняя	Зависит от режима
Уровень автоматизации	Высокий	Высокий	Высокий	Высокий	Средний

Проведённое сравнение показывает, что инструменты SAST и DAST решают различные, но взаимодополняющие задачи. Статический анализ позволяет выявлять уязвимости непосредственно в исходном коде ещё на этапе разработки, тогда как динамическое тестирование ориентировано на обнаружение проблем, возникающих в процессе функционирования приложения. Совместное использование данных инструментов в рамках DevSecOps-конвейеров обеспечивает комплексный подход к обеспечению безопасности программных систем.

Практическая значимость

Практическая значимость проведённого исследования заключается в разработке рекомендаций по интеграции инструментов автоматизированного анализа безопасности в процессы непрерывной интеграции и доставки программного обеспечения. Полученные результаты демонстрируют возможности применения инструментов SAST и DAST для регулярного контроля безопасности программных систем в рамках DevSecOps-подхода. Использование подобных инструментов позволяет выявлять уязвимости на ранних этапах разработки, что существенно снижает вероятность их попадания в производственную среду.

Результаты исследования могут быть использованы при проектировании и внедрении безопасных CI/CD-конвейеров в организациях, занимающихся разработкой программного обеспечения. Практическое применение

предложенного подхода способствует повышению уровня защищённости программных продуктов, снижению рисков эксплуатации уязвимостей и оптимизации процессов обеспечения безопасности в рамках жизненного цикла разработки.

Кроме того, полученные выводы могут быть полезны специалистам в области информационной безопасности и разработчикам программных систем при выборе инструментов анализа безопасности и формировании стратегии внедрения DevSecOps-процессов. Это позволяет повысить эффективность процессов разработки и обеспечить более высокий уровень защиты современных информационных систем.

Заключение

В ходе проведённого исследования были рассмотрены современные подходы к обеспечению безопасности программного обеспечения в условиях использования практик DevOps и CI/CD. Особое внимание было уделено концепции DevSecOps, предполагающей интеграцию механизмов контроля безопасности непосредственно в процессы разработки и доставки программных продуктов. Такой подход позволяет обеспечить непрерывный мониторинг безопасности программных систем на протяжении всего жизненного цикла разработки.

В работе был проведён анализ инструментов статического и динамического тестирования безопасности приложений. Рассмотрены особенности функционирования инструментов SAST и DAST, а также их роль в автоматизированных конвейерах CI/CD. Результаты сравнительного анализа показывают, что статический анализ эффективен для выявления уязвимостей непосредственно в исходном коде, тогда как динамическое тестирование позволяет обнаруживать проблемы безопасности, проявляющиеся в процессе функционирования приложения.

Проведённое исследование подтверждает, что совместное использование инструментов SAST и DAST в рамках DevSecOps-подхода позволяет существенно повысить уровень защищённости программных систем. Интеграция автоматизированных механизмов анализа безопасности в CI/CD-конвейеры обеспечивает раннее обнаружение уязвимостей, снижает риски их попадания в производственную среду и способствует формированию более устойчивых и безопасных программных решений.

Литература

1. OWASP Foundation. OWASP Top 10: The Ten Most Critical Web Application Security Risks [Электронный ресурс]. URL: https://owasp.org/www-project-top-ten/ (дата обращения: 31.03.2026).
2. SonarSource. SonarQube Documentation [Электронный ресурс]. URL: https://docs.sonarsource.com (дата обращения: 31.03.2026).
3. Checkmarx Ltd. Checkmarx Application Security Platform [Электронный ресурс]. URL: https://checkmarx.com (дата обращения: 31.03.2026).
4. OWASP Foundation. OWASP ZAP Project Documentation [Электронный ресурс]. URL: https://www.zaproxy.org (дата обращения: 31.03.2026).
5. Semgrep Inc. Semgrep Documentation [Электронный ресурс]. URL: https://semgrep.dev (дата обращения: 31.03.2026).
6. Shahin M., Babar M.A., Zhu L. Continuous Integration, Delivery and Deployment: A Systematic Review // IEEE Access. 2017. Vol. 5. P. 3909–3943.
7. Rahman M., Williams L. Security Practices in DevOps: A Systematic Mapping Study // Proceedings of the International Conference on Software Engineering. 2019. P. 1–12.
8. Behl A., Behl K. Cybersecurity and Cyberwar: What Everyone Needs to Know. Oxford: Oxford University Press, 2017.

Literature

1. OWASP Foundation. OWASP Top 10: The Ten Most Critical Web Application Security Risks [Electronic resource]. URL: https://owasp.org/www-project-top-ten/ (accessed: 03/31/2026).
2. SonarSource. SonarQube Documentation [Electronic resource]. URL: https://docs.sonarsource.com (accessed: 03/31/2026).

3. Checkmarx Ltd. Checkmarx Application Security Platform [Electronic resource]. URL: https://checkmarx.com (accessed: 03/31/2026).

4. OWASP Foundation. OWASP ZAP Project Documentation [Electronic resource]. URL: https://www.zaproxy.org (accessed: 03/31/2026).

5. Semgrep Inc. Semgrep Documentation [Electronic resource]. URL: https://semgrep.dev (accessed: 03/31/2026).

6. Shahin M., Babar M.A., Zhu L. Continuous Integration, Delivery and Deployment: A Systematic Review // IEEE Access. 2017. Vol. 5. P. 3909–3943.

7. Rahman M., Williams L. Security Practices in DevOps: A Systematic Mapping Study // Proceedings of the International Conference on Software Engineering. 2019. P. 1–12.

8. Behl A., Behl K. Cybersecurity and Cyberwar: What Everyone Needs to Know. Oxford: Oxford University Press, 2017.