

К. Ю. Копыл¹✉

Анализ проблематики автоматизированного аудита Kubernetes-кластеров

¹Национальный исследовательский ядерный университет «МИФИ»,
г. Москва, Российская Федерация
e-mail: kikopyl3@gmail.com

Аннотация. В работе рассматривается проблематика автоматизированного аудита информационной безопасности Kubernetes-кластеров в условиях массового внедрения микросервисной архитектуры. Актуальность темы обусловлена тем, что существующие инструменты сканирования уязвимостей и проверки конфигураций генерируют избыточное количество оповещений, большинство из которых не представляют реальной угрозы в конкретной эксплуатационной среде. Данная проблема определяется как «контекстная слепота» изолированных сканеров, опирающихся исключительно на статические метрики вроде CVSS. Цель исследования – анализ ограничений современных подходов к аудиту и обоснование необходимости перехода к контекстно-ориентированной модели оценки рисков, учитывающей сетевую экспозицию, ролевую модель доступа, чувствительность данных и параметры среды исполнения. В ходе работы выполнена классификация существующих решений по четырем классам (анализаторы конфигураций, сканеры образов, системы runtime, агрегаторы) и проведен их сравнительный анализ по расширенным техническим критериям. Выявлены принципиальные ограничения существующего инструментария, включая ближайший инженерный аналог — утилиту Reticulum. Основным результатом работы является формулирование концептуальных требований к мета-анализатору, агрегирующему данные узкоспециализированных сканеров и выполняющему перерасчет риска на основе метаданных Kubernetes API. Предложенная концепция создает методическую основу для перехода к риск-ориентированной модели управления уязвимостями и определяет направления дальнейших исследований, включая валидацию контекстных коэффициентов методами многокритериального анализа.

Ключевые слова: Kubernetes, контейнеризация, сканирование уязвимостей, контекстно-ориентированный анализ, оценка рисков, DevSecOps

К. У. Копыл¹✉

Analysis of the problems of automated auditing of kubernetes clusters

¹National Research Nuclear University MEPhI, Moscow, Russian Federation
e-mail: kikopyl3@gmail.com

Abstract. This paper examines the problem of automated information security auditing of Kubernetes clusters in the context of widespread adoption of microservice architecture. The relevance of the topic stems from the fact that existing vulnerability scanning and configuration checking tools generate an excessive number of alerts, most of which do not represent a real threat in a specific operational environment. This problem is defined as "context blindness" of isolated scanners that rely solely on static metrics such as CVSS. The aim of the research is to analyze the limitations of modern auditing approaches and to justify the need for a transition to a context-aware risk assessment model that takes into account network exposure, role-based access control, data sensitivity, and execution environment parameters. In the course of the work, a classification of existing solutions into four classes (configuration analyzers, image scanners, runtime systems, and aggregators) was carried out, and their

comparative analysis according to extended technical criteria was performed. Fundamental limitations of the existing toolset were identified, including the closest engineering counterpart – the Reticulum utility. The main result of the work is the formulation of conceptual requirements for a meta-analyzer that aggregates data from specialized scanners and performs risk recalculation based on Kubernetes API metadata. The proposed concept provides a methodological foundation for transitioning to a risk-based vulnerability management model and defines directions for further research, including the validation of context coefficients using multi-criteria analysis methods.

Keywords: Kubernetes, containerization, vulnerability scanning, context-aware analysis, risk assessment, DevSecOps

Введение

В условиях массового перехода корпоративных ИТ-инфраструктур на микросервисную архитектуру платформа оркестрации Kubernetes закрепила статус основного инструмента управления контейнерами. По данным Cloud Native Computing Foundation (2023), его используют 71% компаний Fortune 100 [1]. Однако сложность настройки распределенных компонентов расширяет поверхность атаки и среди наиболее распространенных проблем безопасности Kubernetes выделяются уязвимости приложений (37% организаций), ошибки управления доступом (32%), небезопасные образы (32%) [2]. При этом 45% успешных атак нарушают основную деятельность организаций [3].

Нормативная база (NIST SP 800-190 [4], руководство NSA/CISA, Приказ ФСТЭК №118 [5]) требует регулярного контроля кластеров. Однако внедрение средств автоматизированного сканирования порождает новую эксплуатационную проблему – это избыточность оповещений (alert fatigue). Причина заключается в том, что большинство практических детекторов и сканеров оценивают риск изолированно, опираясь исключительно на статическую базовую метрику CVSS, без учета динамического контекста оркестрируемой среды.

Однако сканирование вызывает проблему избыточности оповещений (alert fatigue) – ИБ-подразделения получают массу отчетов о критических уязвимостях, не несущих реальной угрозы в текущей архитектуре. Причина: сканеры оценивают риск изолированно по статичной метрике CVSS, игнорируя динамический контекст среды.

Целью данного исследования ставится анализ проблематики автоматизированного аудита Kubernetes-кластеров, выявление ограничений существующих подходов и обоснование необходимости перехода к контекстно-ориентированной модели оценки рисков, учитывающей сетевую экспозицию, ролевую модель доступа и параметры среды исполнения.

Сравнительный анализ подходов к аудиту

Для выявления причин неэффективности классического аудита существующие программные решения целесообразно классифицировать на четыре основных группы, каждый из которых имеет специфические ограничения.

К первому классу решений относятся статические анализаторы конфигураций, такие как Kube-bench, Checkov и KubeLinter. Их задача сводится к проверке

YAML-манифестов и параметров запуска узлов на соответствие стандартам, таким как CIS Kubernetes Benchmark. Ограничение подхода заключается в бинарной логике проверок: система фиксирует нарушение (например, отсутствие сетевой политики), но не способна оценить, какой именно код выполняется внутри изолируемого сегмента [6].

Следующую группу образуют сканеры уязвимостей образов, представленные инструментами Trivy, Grype и Clair. Инструменты анализируют слои контейнеров, извлекают спецификации программного обеспечения (SBOM) и ищут совпадения с базами CVE. Главный недостаток метода состоит в отрыве от среды выполнения. Сканер маркирует уязвимую библиотеку высоким приоритетом, не анализируя сетевую связность пода и его полномочия в кластере [7].

Отдельную категорию составляют системы анализа времени выполнения. Решения на базе eBPF (Falco, Tetragon) перехватывают системные вызовы на уровне ядра ОС хоста. Несмотря на высокую точность выявления аномалий (например, попыток запуска интерактивной оболочки), подход носит реактивный характер и неприменим для задач превентивного патч-менеджмента [8].

Наконец, четвертую группу формируют системы агрегации, к которым относят DefectDojo и vimp. Данные платформы консолидируют отчеты сторонних утилит в единой базе данных. Однако на практике агрегация сводится к централизации хранения: уязвимости в коде и параметры доступа пода фиксируются независимо друг от друга, без выстраивания корреляционных связей.

Результаты сравнительного анализа обобщены в табл. 1.

Таблица 1. Сравнение подходов к автоматизированному аудиту

Класс решений	Проверяемые объекты	Базовая метрика риска	Эксплуатационные ограничения
Анализаторы конфигураций	Манифесты, узлы K8s, RBAC	Бинарная (соответствует / не соответствует)	Изоляция от данных об уязвимостях прикладного ПО
Сканеры образов	Пакеты ОС, зависимости кода	Базовый балл CVSS	Отсутствие учета сетевой связности и привилегий
Системы Runtime	Системные вызовы, трафик	Приоритет сигнатуры	Реактивность, невозможность оценки до инцидента
Системы агрегации	Отчеты сторонних утилит	Сводная статистика	Отсутствие перерасчета рисков на основе топологии

Анализ приведенной выше таблицы показывает, что узкая специализация инструментов формирует системную проблему, при которой применение средств по отдельности снижает общую достоверность аудита.

Детальное сравнение инструментов по техническим критериям

Для более глубокого анализа целесообразно перейти от обобщенных классов к сравнению конкретных программных продуктов, отобранных по одному

представителю из каждого функционального класса, а также инструмента, заявляющего о наличии элементов контекстной оценки. В перечень включены: Kube-bench (анализатор конфигураций), Trivy (сканер образов), Falco (Runtime Security), Kubescape (гибридная платформа, сочетающая функции агрегации и оценки соответствия) и Reticulum (утилита контекстной приоритизации уязвимостей).

Таблица 2. Сравнение инструментов аудита по техническим критериям

Инструмент	Модель сбора данных	Учет контекста исполнения	Гибкость настройки приоритетов	Корреляция CVE и событий	Формат и качество отчетов
Kube-bench	Безагентный анализ манифестов и конфигурации узлов через API или локальные файлы	Нет. Бинарная логика проверок (Pass/Fail/Warning), без учета прикладных данных	Фиксированный набор правил CIS	Нет	Консольный вывод и структурированный JSON/текстовый отчет
Trivy	Безагентное сканирование реестров и локальных образов	Минимальный учет контекста: игнорирование CVE по ID или через файл .trivyignore	Наличие .trivyignore	Нет	Поддержка JSON, SARIF и HTML-форматов отчетов
Falco	Агентский анализ через eBPF и модуль ядра	Высокий учет контекста: правила с условиями по namespace, labels, образу и другим параметрам	Пользовательские правила (YAML)	С SIEM/UI через внешние системы	Логи в форматах syslog и JSON, пригодных для интеграции с SIEM
Kubescape	Безагентный анализ через API Kubernetes	Частичный учет контекста: исключение пространств имен, но без динамической топологии и экспозиции	Фреймворки (NSA, MITRE), без гибкой настройки весов	Базовая корреляция реализована в open-source версии. Визуальная корреляция реализована в коммерческой версии	Поддержка JSON, SARIF, HTML и JUnit XML и PDF-форматов отчетов
Reticulum	Безагентный анализ манифестов и данных API Kubernetes	Сетевая доступность (Ingress, тип сервиса) как фактор приоритета	Жестко заданные весовые коэффициенты	Нет	Консольный вывод и структурированный JSON-отчет

В качестве критериев сравнения выбраны параметры, отражающие способность инструмента учитывать эксплуатационный контекст и адаптироваться под нужды конкретной инфраструктуры (табл. 2).

Анализ приведенной выше таблицы показывает, что ни один из рассмотренных инструментов не обеспечивает полноценной корреляции между статическими уязвимостями и динамическим окружением их эксплуатации. Наиболее близким к решению данной задачи является проект Reticulum, который реализует контекстную корректировку приоритета CVE в зависимости от сетевой доступности

сервиса, однако используемые весовые коэффициенты задаются в рамках контекстного анализа уязвимостей без явного формального обоснования. В то же время, в доступных описаниях не найдено свидетельств того, что инструмент учитывает ряд важных факторов безопасности: наличие привилегированного режима у контейнера (`securityContext.privileged`), тип обрабатываемых данных в виде `StatefulSet` с подключенными постоянными томами (PVC), целевое назначение пространства имен (`production / development`) и детализированные ролевые права `ServiceAccount`.

Таким образом, существующий инструментарий, включая *Reticulum*, либо игнорирует контекст, либо реализует его учет в ограниченном и эмпирическом виде, что создает предпосылки для разработки формализованного метода контекстно-ориентированной оценки рисков.

Проблема контекстной слепоты

Неспособность системы аудита учитывать архитектурные и конфигурационные условия функционирования уязвимого компонента в рамках данной работы определяется как «контекстная слепота».

На практике статические метрики риска теряют релевантность в контейнерной среде. В качестве примера можно рассмотреть обнаружение критической уязвимости (оценка CVSS 9.8) в библиотеке `curl` внутри рабочего пода, развернутого в изолированном пространстве имен, не имеющего маршрутизации извне (отсутствие объектов `Ingress`) и ограниченного запрещающими сетевыми политиками. В таких условиях риск практической эксплуатации угрозы является минимальным.

С другой стороны, запуск контейнера с привилегированным доступом в продуктивном контуре представляет собой критическую архитектурную уязвимость, позволяющую злоумышленнику преодолеть границы изоляции и получить доступ к операционной системе сервера, на котором размещен контейнер, либо к другим контейнерам, выполняющимся в той же среде [9, 10].

Концепция контекстно-зависимого аудита

Проведенный анализ показывает, что устранение «контекстной слепоты» невозможно в рамках изолированного применения существующих сканеров. Требуется переход от статической инвентаризации угроз к управляемому корреляционному анализу. В связи с этим предлагается архитектура мета-анализатора, который агрегирует результаты работы узкоспециализированных утилит (например, *Kube-bench*, *Trivy* и других) и выполняет перерасчет уровня риска на основе метаданных, получаемых непосредственно из `Kubernetes-API`. Простейшим интуитивным подходом является введение мультипликативной модели:

$$R = V_{base} \times K_{context} ,$$

где R – единый показатель риска, V_{base} – исходная оценка уязвимости (например, балл CVSS), а $K_{context}$ – композитный коэффициент контекста.

Однако прямое применение этой модели без формального обоснования весовых коэффициентов приводит к ряду нежелательных эффектов и проблем эксплуатации.

Во-первых, числовые значения весов контекста зачастую задаются разработчиками эмпирически, без опоры на статистику инцидентов или формализованные экспертные процедуры.

Во-вторых, введение «презумпции критичности» и присвоение максимальных весов компонентам с неопределенным контекстом (что часто предлагается для предотвращения ложноотрицательных срабатываний) на практике приводит к лавинообразному росту ложноположительных срабатываний.

В качестве третьего вектора проблем можно обозначить отсутствие компенсационных механизмов. Простое перемножение коэффициентов предполагает, что все факторы равнозначны и линейно влияют на итоговый риск. В реальности наличие привилегированного режима является блокирующим фактором, который не может быть компенсирован низкой сетевой экспозицией или отнесением контейнера к тестовой среде.

Таблица 3. Группы контекстных параметров для оценки риска

Группа параметров	Примеры анализируемых данных из Kubernetes API	Характер влияния на риск
Сетевая экспозиция	Ingress, Service (тип LoadBalancer/NodePort), NetworkPolicy	Определяет доступность уязвимого компонента для внешнего атакующего
Привилегии и доступ (IAM)	SecurityContext (privileged, capabilities), ServiceAccount (роли и ClusterRole)	Определяет потенциальный ущерб от компрометации пода (возможность выхода на узел или доступ к секретам)
Чувствительность данных	Наличие PersistentVolumeClaim, монтирование Secrets/ConfigMaps	Влияет на критичность последствий утечки или повреждения данных
Характеристика среды	Namespace (метки prod/dev), пользовательские аннотации	Отражает бизнес-критичность сервиса и допустимый уровень риска

Для преодоления указанных ограничений разработка формализованного метода оценки риска должна опираться на следующую концептуальную схему, включающую четыре группы анализируемых параметров (табл. 3).

Научная задача на следующем этапе исследования заключается не в эвристическом подборе весов, а в выборе и обосновании математического аппарата, адекватного природе угроз в оркестрируемой среде. В качестве перспективных направлений могут быть рассмотрены методы многокритериального принятия решений, в частности метод анализа иерархий (МАИ) [11], а также аппарат нечеткой логики.

Таким образом, предложенная концепция задает вектор перехода к риск-ориентированной модели. Практическая реализация такой модели позволит алгоритмически снижать приоритет уязвимостей, характеризующихся низким уровнем реальной эксплуатационной угрозы в данном архитектурном контексте.

Заключение

Предложенный в работе анализ задает схему сравнительной оценки инструментов аудита Kubernetes-кластеров с точки зрения применимости для контекстной приоритизации угроз. Научная новизна подхода состоит в переходе от изолированной оценки уязвимостей по метрикам CVSS к контекстно-ориентированной модели, учитывающей сетевую экспозицию, ролевую модель доступа, чувствительность данных и параметры среды исполнения. В отличие от существующих инженерных решений, реализующих эвристическую корректировку приоритетов, в работе обоснована необходимость формализованного метода расчета риска на основе метаданных кластера.

Ключевой результат исследования заключается в выявлении проблемы «контекстной слепоты» изолированных сканеров и формулировании концептуальных требований к мета-анализатору, агрегирующему результаты узкоспециализированных утилит. Практическая значимость работы состоит в создании методической основы для перехода к риск-ориентированной модели управления уязвимостями, позволяющей снизить избыточность оповещений.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Kubernetes Project Journey Report [Электронный ресурс] // CNCF. – 2023. – URL: <https://www.cncf.io/reports/kubernetes-project-journey-report/> (дата обращения: 22.10.2025).
2. Red Hat State of Kubernetes Security Report. – 2024.
3. Актуальные киберугрозы: I–II кварталы 2025 года [Электронный ресурс] // Positive Technologies. – 2025. – URL: <https://ptsecurity.com/research/analytics/aktual-nye-kiberugrozy-i-ii-kvartaly-2025-goda/> (дата обращения: 22.10.2026).
4. Souppaya M. Application Container Security Guide / M. Souppaya, J. Morello, K. Scarfone DOI: 10.6028/NIST.SP.800-190. – National Institute of Standards and Technology, 2017.
5. Требования по безопасности информации. Утверждены приказом ФСТЭК России от 4 июля 2022 г. N 118 // ФСТЭК России. – 2022.
6. Денисенко Д.И. Методика тестирования на проникновение контейнерной технологии Kubernetes / Д.И. Денисенко // Информационная безопасность, искусственный интеллект и большие данные : сб. ст. студентов и аспирантов ИИКС. – 2024. – С. 87.
7. Ramasankar Molleti. Container runtime security detection and prevention techniques // World Journal of Advanced Research and Reviews. – 2024. – Vol. 24. – № 3. – P. 2626–2639.
8. Лавров Б.О. Интеллектуальное обнаружение аномалий в контейнеризованных приложениях: методы, архитектура и инструменты / Б.О. Лавров, М.А. Иванов // Безопасность информационных технологий. – 2025. – Т. 32, № 4. – С. 149-164.
9. Agrawal M. Security Audit of Kubernetes based Container Deployments: A Comprehensive Review / M. Agrawal, K. Abhijeet. – 2020. – Vol. 07. – № 06.
10. Липатова С. Е., Белов Ю. С. Практики обеспечения кибербезопасности в Kubernetes // E-Scio. – 2022. – №. 1 (64). – С. 489-497.
11. Корчагин П.В. Модель контроля целостности кластера Linux-серверов под управлением Kubernetes / П.В. Корчагин, А.Д. Шульженко, А.Б. Кириенко // Вопросы оборонной техники. Серия 16: Технические средства противодействия терроризму. – 2023. – № 11-12 (185-186). – С. 16-26.

© К. Ю. Копыл, 2026