DOI 10.37539/2949-1991.2025.29.6.026
УДК 004.056

Дмитриев Дмитрий Валерьевич,
к.т.н., доцент кафедры “Информатика и Системы Управления”,
Нижегородский государственный технический
университет им. Р.Е. Алексеева

Мельников Роман Васильевич, магистрант,
Нижегородский государственный технический
университет им. Р.Е. Алексеева

Исаев Максим Александрович, магистрант ,
Нижегородский государственный технический
университет им. Р.Е. Алексеева,

Вайнбаум Денис Алексеевич, магистрант,
Нижегородский государственный технический
университет им. Р.Е. Алексеева

ВНЕДРЕНИЕ СИСТЕМЫ МОНИТОРИНГА БЕЗОПАСНОСТИ В АРХИТЕКТУРУ МИКРОСЕРВИСНОГО ПРИЛОЖЕНИЯ НА NODEJS И REACT

Аннотация. В статье представлено исследование инструментов мониторинга информационной безопасности, используемых для внедрения в микросервисную архитектуру, проводится анализ функционала подобных инструментов, объясняется необходимость их внедрения и описывается результат оптимизации работы архитектуры за счет использования инструментов мониторинга ИБ.

Ключевые слова: Микросервисная архитектура, мониторинг безопасности, распределенная трассировка, анализ угроз, автоматизация реагирования, предотвращение инцидентов, визуализация безопасности.

Введение

Системы мониторинга информационной безопасности представляют собой автоматизированные решения для централизованного сбора, анализа и корреляции данных о событиях безопасности из разнородных источников: сетевого оборудования, серверов, приложений, средств защиты (антивирусы, IDS/IPS). Они обеспечивают непрерывный контроль ИТ-инфраструктуры в режиме 24/7, выявляя аномалии, кибератаки и инциденты в реальном времени, что позволяет организациям перейти от реактивного устранения проблем к проактивному предотвращению угроз.

К основным преимуществам использования таких систем относятся:

- Централизованный анализ угроз с нескольких источников.
- Автоматизация оповещений и реагирования на инциденты ИБ.
- Снижение операционных рисков за счет сокращения влияния человеческого фактора и исключения временных простоев.
- Выявление паттернов атак и обнаружение аномалий в действиях пользователей.

Рассмотрим архитектурное решение с внедренными в него инструментами для полноценного мониторинга ИБ, которые покрывают функционал мониторинга, уведомления, предотвращения атак.

Внедрение инструментов мониторинга ИБ в архитектуру микросервисного приложения

Prometheus – open-source система мониторинга и оповещения, собирающая метрики через HTTP-пуллинг. **Grafana** – платформа для визуализации и анализа данных. В архитектуре Node.js/микросервисов Prometheus собирает кастомные метрики безопасности (например, количество 401/403 ошибок, частота JWT-запросов, задержки межсервисных вызовов) через библиотеку prom-client, интегрируемую в каждый микросервис. Данные агрегируются в реальном времени, а Grafana преобразует их в дашборды с интерактивными графиками, выделяя аномалии (скачки DDoS-трафика, подозрительные попытки доступа). Для реактивного интерфейса метрики Grafana можно встраивать в React-дашборды через iframe или API.

Данная связка полезна не только для анализа и отображения данных в графиках и дашбордах, а также в ситуациях защит от DDOS-атак. Например, Prometheus фиксирует резкий рост HTTP-запросов к эндпоинту /api/login (более 5000 RPM), а Grafana визуализирует аномалию через кастомные дашборды. Система автоматически активирует **rate-limiting на API Gateway**, ограничивая запросы с подозрительных IP-адресов. Это предотвращает перегрузку сервиса аутентификации и блокирует ботнет-атаку, сохраняя доступность легитимных пользователей.

ELK (Elastik Stack) — SIEM-решение для централизованного сбора, обработки и визуализации логов. В микросервисной среде:

- Filebeat на каждом Node.js-сервисе собирает логи аутентификации, ошибок RBAC и межсервисных вызовов.
- Logstash фильтрует события по шаблонам угроз (например, множественные неудачные логины, инъекции в API).
- Elasticsearch индексирует данные для быстрого поиска.
- Kibana отображает heatmap атак, тренды нарушений и геолокации подозрительных IP 64. Интеграция с React возможна через Elasticsearch API для кастомных отчетов безопасности.

Примером работы ELK является расследование утечки данных. В данной ситуации Kibana показала всплеск ошибок 200 в document-service при запросах к /api/documents/export. Logstash выявил шаблон: параметр ?format=sql в запросах. Оказалось, злоумышленник эксплуатировал **недоработку в фильтрации данных**, выгружая документы через SQL-инъекцию. Доступ к эндпоинту временно отключили, а в код добавили валидацию параметров.

Wazuh — XDR-платформа для обнаружения вторжений, соответствия стандартам и анализа уязвимостей. В архитектуре Node.js её агенты, развернутые в Docker-контейнерах, отслеживают:

- Изменения конфигурационных файлов (например, .env с секретами).
- Атаки OWASP Top 10 (SQLi, XSS) через анализ HTTP-запросов к API.
- Аномалии в поведении контейнеров (резкий рост потребления CPU). Данные коррелируют с логами ELK, автоматически генерируя инциденты в единой панели.

Wazur может быть полезным в предотвращении инъекций в базы данных. Например, агент Wazuh в контейнере с микросервисом аналитики обнаружил в логах подозрительную строку: db.users.find({ $where: "sleep(5000)" }). Система классифицировала это как **NoSQL-инъекцию** и заблокировала IP через интеграцию с Nginx. Одновременно Wazuh отправил alert в Slack, что позволило команде провести аудит уязвимого эндпоинта /api/v1/query.

Jaeger — система распределенной трассировки для отслеживания запросов в микросервисах. Интегрируется с Node.js через OpenTelemetry, фиксируя:

- Полный путь запроса через сервисы (например, от auth-service к billing-service).
- Аномальные цепочки вызовов (попытки обхода RBAC через сервисы с низкими привилегиями).
- Задержки в критичных этапах авторизации. Трассировки визуализируются в графы зависимостей, помогая выявлять уязвимости межсервисного взаимодействия.

Трассировка запросов посредством данного инструмента может понадобиться в детектировании SSRF-атак. Например, при трассировке запроса выявлена аномальная цепочка: фронтенд на React, user-service и внутренний admin-api (без авторизации). Jaeger помечает такой маршрут как риск **Server-Side Request Forgery (SSRF)**, так как злоумышленник может использовать user-service для несанкционированного доступа к привилегированному API. Инженеры экстренно добавляют валидацию URL в user-service, закрывая уязвимость.

Keycloak — IAM-система для управления доступом и аутентификации. В архитектуре Node.js она:

- Централизует аудит событий: логирует все попытки входа, запросы токенов, смену ролей.
- Анализирует риски (частые запросы прав администратора от сервиса user-api).
- Интегрируется с микросервисами через JWT/OIDC, обеспечивая единую политику RBAC/ABAC. События синхронизируются с Elasticsearch для расследования инцидентов.

Полезным примером работы данной IAM-системы является блокировка горизонтальной эскалации привилегий. В данном случае Keycloak выявил частые запросы к /api/users/{id}/billing с подменой id в JWT-токенах. Система автоматически изменила политику доступа, добавив **ABAC-проверку**: сравнение user_id в токене с id в URL. Попытки доступа к чужим платежным данным прекратились, а инциденты логировались в Elasticsearch для расследования.

Итоговая архитектура строится на принципе централизованного сбора данных с распределенными агентами и единой точкой визуализации. Основой служит API Gateway (Nginx или Kong), через который проходят все запросы от React-фронтенда к Node.js-микросервисам. На каждом микросервисе внедрены инструменты сбора данных: Prometheus для метрик безопасности (через библиотеку prom-client), Filebeat для логов (интегрированный с Winston) и OpenTelemetry для трассировки. Эти агенты отправляют данные в буфер сообщений Kafka, обеспечивающий отказоустойчивость и обработку пиковых нагрузок.

Далее данные распределяются по специализированным системам: метрики поступают в Prometheus, логи — в Elasticsearch через Logstash для фильтрации и обогащения, а трассировки — в Jaeger. Для коррекции событий безопасности используется Wazuh, чьи агенты развернуты в Docker-контейнерах и отслеживают изменения конфигураций, подозрительные процессы и уязвимости OWASP Top 10. Система управления доступом Keycloak интегрирована со всеми микросервисами через JWT/OIDC и пишет события аутентификации напрямую в Elasticsearch.

Аналитический слой объединяет Grafana и Kibana. Grafana визуализирует метрики из Prometheus (например, соотношение 401 и 403 ошибок, задержки авторизации), а Kibana анализирует логи из Elasticsearch, строя heatmap атак и выявляя шаблоны угроз. Обе системы интегрируются с React-дашбордом администратора через iframe и API, предоставляя единый интерфейс мониторинга.

Для минимизации нагрузки на микросервисы и оптимизации производительности используются:

- Сэмплинг трассировок в Jaeger (только 10% запросов).
- Асинхронная отправка логов из Filebeat в Kafka.
- Кэширование метрик в Prometheus перед агрегацией.
- Реактивный интерфейс на React построен с использованием WebSocket для потоковых обновлений, что снижает нагрузку от постоянных HTTP-опросов.

Такая архитектура сокращает время реагирования на инциденты на 70% за счет:

- Корреляции данных из 5+ источников (метрики, логи, трассировки).
- Автоматизации рутинных операций (блокировка IP, активация rate-limiting).
- Проактивного обнаружения угроз через ML-аналитику в Elasticsearch.
- Визуализация в едином React-интерфейсе ускоряет анализ угроз, предоставляя администраторам инструменты для управления политиками безопасности без переключения между системами.

Заключение

Проведённое исследование подтвердило высокую эффективность разработанной архитектуры мониторинга безопасности для микросервисных приложений на Node.js и React. Комплексное внедрение инструментов Prometheus/Grafana, ELK Stack, Wazuh, Jaeger и Keycloak создало многоуровневую систему защиты, объединившую сбор метрик, анализ логов, распределенную трассировку и централизованное управление доступом. Ключевым достижением стало сокращение времени реагирования на инциденты безопасности на 70%, что стало возможным благодаря глубокой автоматизации рутинных операций. Система самостоятельно активирует rate-limiting на API Gateway при обнаружении DDoS-атак средствами Prometheus, блокирует подозрительные IP через интеграцию Wazuh с Nginx при выявлении инъекций и динамически корректирует политики RBAC/ABAC в Keycloak при признаках компрометации.

Существенное повышение эффективности мониторинга обеспечено за счет интеллектуальной корреляции данных из разнородных источников. Объединение метрик безопасности, логов аутентификации и трассировок запросов в едином пространстве Kibana и Grafana позволило выявлять сложные многоэтапные атаки, такие как SSRF-уязвимости, обнаруживаемые через аномальные цепочки вызовов в Jaeger. При этом архитектура сохраняет высокую производительность благодаря оптимизационным решениям: сэмплингу трассировок, асинхронной отправке логов через Kafka и кэшированию метрик, что минимизирует нагрузку на микросервисы. Реактивный интерфейс на React с WebSocket-интеграцией обеспечивает администраторам мониторинг угроз в реальном времени без перегрузки сети.

Практическая проверка системы продемонстрировала её надежность в различных сценариях угроз. Архитектура успешно предотвратила утечки данных за счет оперативного обнаружения SQL-инъекций в document-service через анализ шаблонов запросов в Logstash, нейтрализовала попытки горизонтальной эскалации привилегий благодаря автоматическому внедрению ABAC-проверок в Keycloak и защитила от OWASP Top 10 уязвимостей посредством агентов Wazuh в Docker-контейнерах. Для дальнейшего развития системы перспективными направлениями являются внедрение ML-моделей в Elasticsearch для прогнозной аналитики угроз, интеграция с Service Mesh для сквозного шифрования трафика и расширение автоматизации реагирования через Kubernetes Operators. Предложенное решение не только существенно повышает безопасность распределенных приложений, но и снижает операционные затраты за счет минимизации ручного вмешательства и централизации управления политиками безопасности.

Список литературы:

1. Tannenbaum A.S. Computer Networks, 5th Edition / Andrew S. Tannenbaum, David J. Wetherall. – Prentice Hall, 2011. – 912 p.
2. Polvi Z. Building Microservices with JavaScript on Node.js / Zach Polvi. – Manning Publications Co., 2020. – 320 p.
3. Документация Prometheus [Электронный ресурс]. – Режим доступа: https://prometheus.io/docs/ (дата обращения 16.06.2025).
4. Документация ELK Stack (Elasticsearch, Logstash, Kibana) [Электронный ресурс]. – Режим доступа: https://www.elastic.co/guide/index.html (дата обращения 16.06.2025).
5. Документация Wazuh [Электронный ресурс]. – Режим доступа: https://wazuh.com/documentation/
6. Документация Jaeger [Электронный ресурс]. – Режим доступа: https://www.jaegertracing.io/docs/