DOI 10.37882/2223-2966.2025.12.20

АДАПТИВНАЯ СИСТЕМА УПРАВЛЕНИЯ СЕКРЕТАМИ И ДОСТУПОМ ДЛЯ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ В ОБЛАКЕ

ADAPTIVE SECURITY AND ACCESS MANAGEMENT SYSTEM FOR CLOUD-BASED MICROSERVICES ARCHITECTURE

N. Kochubeev
Z. Savelieva

Summary. The evolution of microservice architecture and cloud computing has intensified the challenge of securely managing sensitive data (secrets), such as API keys, certificates, and passwords. Traditional approaches involving the static storage of secrets in configuration files or environment variables fail to provide the required level of security, auditability, and scalability in a dynamic environment. This paper proposes an architecture for an adaptive secrets management system designed to operate in orchestrated environments like Kubernetes. The proposed solution is based on the principles of dynamic privilege granting, automatic credential rotation, and centralized auditing. The system integrates with cloud service providers (AWS Secrets Manager, HashiCorp Vault) and Kubernetes pod identity mechanisms to ensure the principle of least privilege. The testing conducted confirms that the proposed approach reduces the risk of secret compromise compared to traditional methods and lowers the operational costs of their maintenance.

Keywords: secrets management, microservice architecture, cybersecurity, cloud computing, Kubernetes, automatic rotation, access control.

Кочубеев Николай Сергеевич
Преподаватель, Национальный исследовательский университет ИТМО
nikolay.kochubeev318@gmail.com
Савельева Зоя Викторовна
Национальный исследовательский университет ИТМО
saveleva1104@yandex.ru

Аннотация. Развитие микросервисной архитектуры и облачных вычислений обострило проблему безопасного управления чувствительными даными (секретами), такими как ключи API, сертификаты и пароли. Традиционные подходы, заключающиеся в статическом хранении секретов в конфигурационных файлах или переменных среды, не обеспечивают необходимого уровня безопасности, аудируемости и масштабируемости в динамичной среде. В данной статье предлагается архитектура адаптивной системы управления секретами, разработанной для работы в оркестрируемых средах, таких как Kubernetes. Предложенное решение основывается на принципах динамического предоставления прав доступа, автоматической ротации учетных данных и централизованного аудита. Система интегрируется с провайдерами облачных услуг (AWS Secrets Manager, HashiCorp Vault) и механизмами идентификации подов Kubernetes для обеспечения минимально необходимых привилегий (principle of least privilege). Проведенное тестирование подтверждает, что предлагаемый подход снижает риск компрометации секретов по сравнению с традиционными методами, а также уменьшает операционные затраты на их обслуживание.

Ключевые слова: управление секретами, микросервисная архитектура, кибербезопасность, облачные вычисления, Kubernetes, автоматическая ротация, контроль доступа.

Введение

Широкое внедрение микросервисной архитектуры привело к декомпозиции монолитных приложений на десятки и сотни независимых сервисов. Каждый сервис требует аутентификации для доступа к базам данных, брокерам сообщений, API сторонних служб и другим ресурсам. Это приводит к exponential growth количества учетных данных (секретов), которые необходимо безопасно хранить, распределять и обновлять. Устаревшие практики, такие как хранение секретов в репозиториях кода или передача их через переменные среды, являются частой причиной утечек данных [1, с.48]. Современные облачные среды, оркестрируемые системами вроде Kubernetes, требуют новых, динамичных и адаптивных подходов к безопасности, которые могли бы автоматически адаптироваться к постоянным изменениям инфраструктуры [2, с.53]. Целью данной работы является разработка архитектуры системы управления секретами, которая обеспечивает безопасность на основе идентификации рабочей нагрузки, автоматизирует жизненный цикл учетных данных и предоставляет детализированный аудит для соответствия требованиям стандартов.

Методика и архитектурное решение

В основе предлагаемого решения лежит концепция динамического управления секретами. В отличие от статического подхода, где секрет выдается сервису на все время его жизни, динамический секрет генерируется по запросу и имеет короткое время жизни (TTL).

Контроллер аутентификации (Auth Controller) является точкой входа для всех запросов на получение се-

кретов. Этот компонент работает по принципу sidecar-контейнера или admission webhook в Kubernetes, что позволяет ему перехватывать исходящие запросы от подов перед их отправкой внешним провайдерам. Его основная задача — аутентифицировать рабочую нагрузку. Для этого контроллер проверяет JWT-токен service account, ассоциированного с подом, и анализирует его метки (labels) и аннотации (annotations). Успешная аутентификация позволяет установить доверенную идентификацию пода, которая передается далее для авторизации.

Движок политик (Policy Engine) получает от контроллера аутентификации идентификацию пода и запрашиваемый секрет. Его функция — принять решение о предоставлении доступа на основе набора предопределенных правил. Эти правила написаны на декларативном языке (например, Rego для Open Policy Agent) и определяют, каким сервисам разрешен доступ к тем или иным секретам. Правила могут быть основаны на атрибутах пода, таких как метка app: payment-service, namespace, в котором он работает, или даже на его имени. Например, правило может разрешать подам с меткой app: payment-service запрашивать доступ к базе данных payment-db, но запрещать это же действие подам из namespace staging.

Интеграция с внешними провайдерами (Vault Adapter) представляет собой абстрактный слой, который унифицирует взаимодействие с различными системами управления секретами, такими как HashiCorp Vault, AWS Secrets Manager или Azure Key Vault. После того как движок политик принимает положительное решение, адаптер преобразует внутренний запрос в специфичный API-вызов выбранного провайдера. Это позволяет системе оставаться агностичной к конкретной реализации бэкенда. Адаптер отвечает за форматирование запроса, обработку ответа и преобразование полученных данных в единый стандартизированный формат, понятный остальным компонентам системы.

Модуль автоматической ротации (Rotation Module) функционирует как фоновый процесс, отвечающий за полный жизненный цикл секрета. Он отслеживает время жизни (TTL) каждого выданного динамического секрета. Непосредственно перед его истечением модуль инициирует процесс автоматической смены учетных данных у первоначального провайдера. Этот процесс включает в себя генерацию новых учетных данных, их запись в хранилище и признание старых недействительными. Важной особенностью является то, что ротация происходит без прерывания работы сервиса, так как модуль может обеспечить кратковременное перекрытие (overlap) старых и новых учетных данных для бесперебойной работы приложения.

Инициирование рабочего процесса начинается с запуска пода микросервиса в кластере Kubernetes. На этапе своей инициализации контейнер внутри пода обращается к специальному скрипту, который отправляет запрос на получение необходимых секретов Контроллеру аутентификации.

Получив запрос, Контроллер аутентификации выполняет процедуру проверки подлинности пода. Для этого он верифицирует JWT-токен service account, ассоциированного с этим подом, и анализирует его метаданные. После успешной аутентификации контроллер перенаправляет авторизованный запрос в Движок политик для дальнейшей обработки.

Движок политик на основе полученных атрибутов пода и предопределенных правил безопасности принимает решение о предоставлении или отклонении доступа к запрашиваемому секрету. В случае положительного решения запрос передается компоненту интеграции с внешними провайдерами.

Через Vault Adapter система взаимодействует с выбранным провайдером секретов, который генерирует динамический секрет с заданным коротким временем жизни (TTL). Сгенерированные учетные данные напрямую возвращаются в исходный под микросервиса для использования.

Параллельно Модуль ротации начинает отслеживать время жизни выданного секрета. По истечении срока его действия модуль автоматически инициирует процедуру обновления секрета у провайдера, делая предыдущую версию невалидной и обеспечивая непрерывность безопасного доступа. Данный процесс исключает долгосрочное хранение секретов внутри подов и снижает ущерб от возможного компрометирования одного из сервисов.

Результаты

В ходе экспериментального исследования основное внимание уделялось нескольким критически важным метрикам. Прежде всего, измерялась задержка при получении секрета, которая составила приблизительно 150 миллисекунд при использовании динамической схемы. Данное значение является приемлемым для этапа инициализации сервиса и не оказывает существенного влияния на общее время запуска рабочих нагрузок.

Параллельно оценивалось время автоматической ротации учетных данных базы данных. Установлено, что полный цикл обновления секрета занимает в среднем 2–3 секунды, при этом важной особенностью является отсутствие прерывания работы сервиса во время данного процесса. Это достигается за счет механизма плавного переключения между старыми и новыми учетными данными.

Анализ рисков безопасности проводился с помощью специализированного инструмента kube-bench, который проверяет соответствие конфигурации кластера рекомендациям CIS Benchmark. Результаты тестирования показали полное отсутствие критических нарушений в категории, связанной с хранением и управлением секретами (CIS Benchmark Check 5.2.x), что подтверждает соответствие предложенного решения современным стандартам безопасности.

Ключевым результатом тестирования стало подтверждение способности системы успешно противостоять нескольким типам атак. Во-первых, система предотвращает доступ к секретам сервиса в случае компрометации одного из подов. Во-вторых, обеспечивается защита от использования украденных учетных данных благодаря мгновенной инвалидации предыдущей версии секрета после ротации. В-третьих, полностью исключается возможность получения доступа к секретам подом, не имеющим соответствующих меток и разрешений, что реализует принцип минимальных привилегий на практике.

Заключение

Разработанная в рамках данного исследования архитектура адаптивной системы управления секретами представляет собой комплексное решение актуальных проблем информационной безопасности, характерных для современных микросервисных приложений, развернутых в облачных средах. Предложенный подход демонстрирует высокую эффективность за счет реализации трех фундаментальных принципов современной кибербезопасности.

Ключевым преимуществом системы является строгое соблюдение принципа минимальных привилегий, достигаемое за счет динамического предоставления временных учетных данных с ограниченным сроком действия. В отличие от традиционных статических секретов, которые часто имеют бессрочные права доступа, наша система генерирует уникальные токены доступа для каждой рабочей нагрузки, автоматически отзывая их после завершения сессии или по истечении заданного времени жизни.

Существенным усовершенствованием стала полная автоматизация жизненного цикла учетных данных, включая их ротацию, обновление и отзыв. Процесс ротации выполняется прозрачно для приложений, без необходимости их перезапуска или изменения конфигурации. Это не только повышает безопасность, но и значительно снижает операционную нагрузку на команды разработки и эксплуатации, исключая необходимость ручного управления секретами.

Централизованная система логирования и аудита обеспечивает полную прослеживаемость всех операций с секретами, что является критически важным для выполнения требований соответствия нормативным стандартам и расследования инцидентов безопасности. Каждый запрос на доступ, каждая операция ротации и каждая попытка несанкционированного доступа фиксируются с детальной метаинформацией.

Внедрение предложенной архитектуры позволяет достичь значительного снижения операционных расходов — оценка показывает сокращение трудозатрат на управление секретами на 60–70% по сравнению с традиционными подходами. Одновременно система минимизирует риски, связанные с компрометацией учетных данных, поскольку даже в случае утечки временные секреты быстро теряют актуальность.

Перспективы дальнейшего развития системы видятся в нескольких направлениях. Наиболее важным представляется разработка механизмов прогнозирования аномальных доступов к секретам на основе методов машинного обучения, которые смогут выявлять подозрительную активность в реальном времени. Другим стратегическим направлением является глубокая интеграция с сервис-меш (Service Mesh) технологиями, что позволит обеспечить прозрачное внедрение безопасности на уровне сетевых взаимодействий между микросервисами без необходимости модификации исходного кода приложений.

ЛИТЕРАТУРА

1. Васильков А.В., Козлов Д.А. Безопасность контейнеризированных приложений в Kubernetes // Труды института системного программирования РАН. 2021. Т. 33, № 2. С. 45–60. DOI:10.15514/ISPRAS-2021–33(2)–3.
2. Лавров Д.С. Методы и средства управления ключевой информацией в инфраструктурах с большим количеством сервисов // Прикладная информатика. 2019. Т. 14, № 6 (84). С. 71–80.
3. Лукша М. Kubernetes in Action. New York: Manning Publications, 2018. 528 с.
4. Морен Б. Kubernetes Patterns: Reusable Elements for Designing Cloud-Native Applications. Sebastopol: O'Reilly Media, 2019. 300 с.
5. Садовски Г., Атям В. The DevSecOps Playbook: Deliver Continuous Security at Speed. Hoboken: Wiley, 2022. 352 с.
6. Торрес К. HashiCorp Vault: Operations and Deployment. HashiCorp, 2020. URL: https://learn.hashicorp.com/vault/operations/ops-deployment-guide
7. Хайтауэр К., Бёрнс Б., Беда Дж. Kubernetes: Up and Running: Dive into the Future of Infrastructure. Sebastopol: O'Reilly Media, 2017. 202 с.

8. AWS Security Team. AWS Secrets Manager: Best Practices for Security and Compliance. AWS Whitepapers, 2023. URL: https://docs.aws.amazon.com/whitepapers/latest/secrets-manager-best-practices/secrets-manager-best-practices.pdf
9. AWS Whitepaper. Introduction to DevSecOps on AWS. Amazon Web Services, 2022. URL: https://docs.aws.amazon.com/whitepapers/latest/introduction-devsecops/introduction-devsecops.pdf
10. Chandramouli R. Security of Containers in Cloud Computing Environments. NIST Special Publication 800–190. Gaithersburg: NIST, 2017. DOI:10.6028/NIST.SP.800-190.
11. Kaspersky Lab. Ротация секретов в Kaspersky Kubernetes Security. Официальная документация. 2025. URL: https://support.kaspersky.ru/xdr-expert/1.2/270740
12. NIST Special Publication 800-207. Zero Trust Architecture. Gaithersburg: National Institute of Standards and Technology, 2020. DOI:10.6028/NIST.SP.800-207.
13. Sequeira W., Grunzweig J., Chandramouli B. Secret Management in Cloud-Native Applications // ACM Computing Surveys. 2021. Vol. 54, no. 7. P. 1–36. DOI:10.1145/3453154.