Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Разработка компонентов аутентификации, авторизации и учета в распределенных системах

Санкт-Петербург
2025

Выполнил: студент гр. 5130903/20302 Н.Ю. Каралюс,
Руководитель: доцент Сергеев А.В

Платформа

Мы создаём собственное **PaaS решение**, которое упростит жизнь разработчиков и позволит малым компаниям не тратить ресурсы на платформенные команды.

- Деплой в Kubernetes и VM в пару кликов – мощный функционал без сложной настройки.
- GitOps-first подход – агенты сами читают репозитории, никаких ручных пайплайнов.
- Безопасность по умолчанию – защищённое хранилище секретов, аудит действий и автоматическая ротация.
- Полная observability из коробки – метрики, логи, трейсы и быстрый поиск причин инцидентов.
- Мониторинг, OnCall, дежурства и ownership команд – всё готово к продакшену.
- Progressive delivery и авто-роллбеки – безопасные релизы без страха.
- Контроль затрат – расходы по сервисам и командам, лимиты и алерты.

Фокус данной работы – безопасность.

Актуальность

- Распространение микросервисной архитектуры и облачной оркестрации.
- Расширение поверхности атаки при дроблении приложений на множество компонентов.
- Увеличение сложности управления безопасностью в распределенных системах.
- Необходимость обеспечения корректной идентификации и разграничения прав для пользователей и сервисов.
- Низкая степень унификации и ограниченная доступность готовых Open Source решений, полностью покрывающих требования по AAA.
- **Платформа будет иметь доступ с высокими привилегиями к критической инфраструктуре клиента. Если платформа не может обеспечить безопасность, то такая платформа неконкурентоспособна.**

Практическая значимость

- AAA-подсистема обеспечит решение практических задач безопасности и аудита в экосистеме платформы.
- Платформа получит **продуктовое решение по безопасности**, которое позволит быстрее развиваться.
- Вклад в развитие подходов к обеспечению безопасности в распределенных системах.

Цель

Разработка и реализация компонентов подсистемы аутентификации, авторизации и учета (далее AAA) для распределенной микросервисной платформы, обеспечивающей безопасное взаимодействие пользователей и сервисов.

Задачи

1. Изучить современные подходы и механизмы обеспечения безопасности в распределенных микросервисных системах.
2. Спроектировать архитектуру AAA-подсистемы, включающую компоненты пользовательской аутентификации и авторизации, межсервисной аутентификации и централизованного аудита.
3. Реализовать компоненты пользовательской аутентификации и авторизации.
4. Реализовать компоненты межсервисной аутентификации и авторизации.
5. Реализовать компоненты аудита и мониторинга.
6. Провести апробацию разработанной AAA-подсистемы.

Анализ литературы

Межсервисная аутентификация

- mTLS является предпочтительным механизмом для межсервисной коммуникации в Kubernetes.
- Service Mesh предоставляет встроенную поддержку mTLS без изменения кода приложений.
- Для сервисов с персональными данными mTLS является обязательным требованием.

Пользовательская аутентификация

- OAuth 2.0 и JWT остаются доминирующими способами аутентификации и авторизации.
- Необходимость интеграции с Identity Provider.
- Поддержка Single Sign-On критична для пользовательского опыта.

Логирование и аудит

- Рекомендуется паттерн с агентом-сборщиком логов.
- Логи публикуются в message broker в HA кластере.
- Продуктовые логи – это отдельный домен и сервис в системе.

Анализ литературы

Управление секретами

- Секреты должны хранится в защищенных хранилищах (например, Vault).
- Секреты в Kubernetes должны быть зашифрованы.
- Необходима автоматическая ротация ключей и сертификатов.

Zero Trust Architecture

- Политики inter-pod коммуникации в Kubernetes
- Защита инфраструктура кластера как приоритет (etcd, k8s API, ноды).

Анализ инструментов

1. Identity Provider

	Keycloack	Duende IdentityServer	Authentik	Dex	Ory Hydra
Open Source лицензия	+ Apache 2.0	+- (> $1M revenue – платно)	+ MIT	+ Apache 2.0	+- Apache 2.0 (важная функциональность в платной подписке)
OAuth 2.0 / OIDC	+	+	+	+	+
Встроенное управление пользователями	+	+- (через Identity в ASP.NET)	+	-	-
Нативный Helm Chart	+	+- (возможен, но требует кастомизации)	+	+	+
Кастомизация UI	+	++ (полный контроль через Blazor, последнее – больше минус)	+	-	-
Федерация с внешними IdP	+	+	+	+	+
Fine-grained авторизация	+	+-	+-	-	-
Встроенная админ-панель	+	+-	+	-	-
Поддержка мультитенантности	+	+	+-	-	+-
Документация	++	++	+	+	+
Потребление ресурсов	Высокое (из-за JVM)	Низкое	Среднее	Низкое	Очень низкое (golang)
Гибкость кастомизации логики	+- (через плагины на Java)	++ (полный контроль кода)	+-	+-	+-
Звезды в GitHub	31000	1600	19000	10400	16700

Анализ инструментов

2. Service Mesh

	Istio	Linkerd	Cilium	Consul Connect
Open Source лицензия	+ Apache 2.0	+ Apache 2.0	+ Apache 2.0	+ MPL 2.0
Автоматический mTLS	+	+	+	+
JWT-валидация на уровне mesh	+	-	+-	+-
Authorization Policies	++	+-	+	+
Потребление ресурсов	Высокое	Низкое	Среднее	Среднее
Сложность установки	Высокая	Низкая	Средняя	Средняя
Observability	++	+	+	+
Traffic management	++	+-	+	+
Multi-cluster поддержка	+	+	+	+
Документация	++	++	+	+

3. Observability

	Loki + Promtail	ELK Stack	ClickHouse + Vector	Fluentd + Opensearch
Open Source лицензия	+ AGPL 3.0	+- Elastic License 2.0	+ Apache 2.0	+ Apache 2.0
Потребление ресурсов	Низкое	Высокое	Среднее	Среднее
Full-text поиск	+-	++	+-	+
Индексирование по меткам	++	+	+	+
Нативный Helm Chart	+	+	+	+
Интеграция с Grafana	++	+	+	+
Стоимость хранения	Низкая	Высокая	Низкая	Средняя
Сложность эксплуатации	Низкая	Высокая	Средняя	Средняя
Алерты в реальном времени	+	+	+	+

Анализ инструментов

4. Хранилище секретов

	OpenBao	K8s secrets	CyberArk Conjur
Open Source лицензия	+ MPL 2.0	+ Apache 2.0	+ Apache 2.0
Динамическая генерация секретов	+	-	+
Автоматическая ротация	+	-	+
Аудит доступа к секретам	+	-	+
Kubernetes интеграция	+	++ (это и есть kubernetes)	+
Шифрование	+	+- (нужна настройка etcd)	+
Сложность эксплуатации	Высокая	Низкая	Высокая
High Availability	+	+ (за счет etcd)	+
Федерация	++	+-	+

5. Язык программирования + фреймворк

	Go + stdlib	C# + ASP.NET	Rust + Axum	Java + Sprint Boot	TS + NestJS
Производительность	++	++	+++	+	+
Потребление памяти	Низкое	Низкое	Очень низкое	Высокое	Среднее
Время компиляции	Быстрое	Быстрое	Медленное	Среднее	Быстрое
Kubernetes Client SDK	++	+	+	+	+
OAuth / OIDC библиотеки	+	++	+	+	+
Static Binary	+	- (он есть, но не экспериментальный)	+	-	-
Популярность в CNCF	+++	+-	+	+	+-
Memory Safety	GC	GC	Compile Time	GC	GC
Конкурентность	Goroutines	async/await + Threads	async / await	Threads / Virtual Threads	Event Loop
DevEx	+	++	+-	+-	+
Сложность обучения	Средняя	Средняя	Высокая	Средняя	Низкая

Гипотеза

Для обеспечения безопасности распределенной микросервисной платформы в Kubernetes оптимальным решением является комбинированный подход:

Межсервисная аутентификация

- Istio Service Mesh для автоматического mTLS.
- Автоматическое управление сертификатами через встроенные механизмы Service Mesh.
- Политики авторизации на уровне Service Mesh для fine-grained контроля доступа

Пользовательская аутентификация

- Keycloak как центральный Identity Provider.
- Интеграция через OAuth 2.0/OIDC протоколы.
- JWT токены для передачи контекста авторизации.

Логирование и аудит

- Архитектура с sidecar-агентами на основе Fluent Bit для сбора логов.
- Централизованное хранение в Loki с визуализацией через Grafana.
- Структурированные логи с обязательным включением CorrelationID и категоризацией событий аудита.

Статус работ

Содержание работ:

1. ✅ Определение предметной области, целей и задач системы аутентификации и взаимодействия микросервисов в Kubernetes.
2. ✅ Анализ архитектурных особенностей микросервисных систем и принципов их развертывания в Kubernetes.
3. ✅ Исследование механизмов внутреннего взаимодействия микросервисов и методов безопасной коммуникации (Service Account, mTLS, Service Mesh).
4. ✅ Анализ протоколов и подходов к аутентификации сервисов (API Token, OAuth 2.0, OpenID Connect, JWT).
5. ✅ Исследование способов внешней аутентификации запросов от сторонних систем.
6. ✅ Анализ методов пользовательской авторизации через внешних провайдеров.
7. ✅ Исследование подходов к реализации систем аудита действий пользователя в распределенных приложениях
8. ✅ Систематизация результатов анализа и формирование требований к архитектуре подсистем авторизации и аудита.
9. ⚠️ Формирование плана дальнейших экспериментов и сценариев оценки эффективности предложенного решения.
10. ❌ Составление отчета по результатам теоретического исследования.

Спасибо за внимание!