**MAKALAH**

**MICROSERVICES**

**“KEAMANAN DALAM MICROSERVICES”**



**DISUSUN OLEH:**

**TRI RAHMA AGUSTIN**

**2301082018**

**PROGRAM STUDI TEKNIK KOMPUTER**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI PADANG**

**2024/2025**

# BAB I: PENDAHULUAN

## Latar Belakang

Perkembangan teknologi informasi telah membawa perubahan besar dalam cara pengembangan dan pengelolaan sistem perangkat lunak. Salah satu pendekatan arsitektur modern yang kini banyak digunakan adalah microservice. Microservice memungkinkan pengembangan aplikasi sebagai kumpulan layanan kecil yang saling berinteraksi, masing-masing memiliki tanggung jawab spesifik dan dapat dikembangkan serta di-deploy secara independen. Pendekatan ini memberikan fleksibilitas, skalabilitas, dan kecepatan dalam pengembangan perangkat lunak.

Namun, di balik keunggulan tersebut, arsitektur microservice membawa tantangan tersendiri dalam hal keamanan. Jika dibandingkan dengan arsitektur monolitik, microservice memiliki lebih banyak titik interaksi antar komponen yang tersebar, sehingga meningkatkan potensi risiko keamanan. Setiap layanan microservice berkomunikasi melalui jaringan, yang membuka kemungkinan serangan seperti intersepsi data, spoofing, dan injection attack. Oleh karena itu, keamanan menjadi aspek yang sangat penting dan kompleks dalam penerapan arsitektur microservice.

## 1.2 Rumusan Masalah

1. Apa saja tantangan dan risiko keamanan yang dihadapi dalam arsitektur microservice?
2. Strategi atau praktik terbaik apa yang dapat diterapkan untuk mengamankan microservice?
3. Bagaimana praktik terbaik industri dalam mengimplementasikan keamanan pada microservice?

**1.3 Tujuan Penulisan**

1. Mengidentifikasi tantangan keamanan dan risiko yang umum terjadi pada microservice.
2. Menjelaskan pendekatan dan solusi untuk mengatasi tantangan tersebut.
3. Memberikan rekomendasi berdasarkan penelitian ilmiah dan praktik industri.

# BAB II: TINJAUAN PUSTAKA

## 2.1 Definisi Microservice

Microservice adalah pendekatan arsitektur perangkat lunak yang menyusun aplikasi sebagai kumpulan layanan kecil (service) yang dikembangkan dan dioperasikan secara independen. Setiap layanan biasanya mewakili fungsi bisnis tertentu, memiliki basis data sendiri, dan berkomunikasi dengan layanan lain menggunakan protokol ringan seperti HTTP/REST atau messaging queue seperti Kafka dan RabbitMQ.

## 2.2 Keamanan Sistem Terdistribusi

Microservice merupakan bentuk sistem terdistribusi, yang memiliki tantangan keamanan tersendiri dibandingkan sistem monolitik. Sistem terdistribusi rawan terhadap ancaman seperti intersepsi komunikasi antar layanan, pengambilan alih identitas, dan penggunaan resource oleh pihak tidak berwenang. Oleh karena itu, pendekatan keamanan yang digunakan pun harus mampu menjangkau setiap titik interaksi dalam sistem tersebut.

## 2.3 Studi Sebelumnya

Berdasarkan penelitian oleh Di Francesco et al. (2021), banyak organisasi mengalami kesulitan dalam mengamankan microservice karena kurangnya standar dan tooling yang memadai. Studi oleh Taibi et al. (2020) juga mencatat bahwa meskipun adopsi microservice meningkat, praktik keamanannya belum berkembang secepat teknologinya.

**2.4 Karakteristik Microservice**

* **Decentralized Governance:** Pengelolaan konfigurasi, database, dan deployment dilakukan secara terdesentralisasi.
* **Technology Diversity:** Setiap layanan bisa dikembangkan dengan bahasa pemrograman atau framework yang berbeda.
* **Resilience:** Microservice dirancang agar kegagalan satu layanan tidak berdampak ke seluruh sistem.
* **Scalability:** Layanan dapat diskalakan secara independen berdasarkan kebutuhan.

# BAB III: PEMBAHASAN

## 3.1 Tantangan Keamanan dalam Microservice

1. Permukaan Serangan yang Luas: Banyaknya layanan dan API meningkatkan titik masuk bagi penyerang.
2. Manajemen Identitas dan Akses: Setiap layanan membutuhkan sistem otentikasi dan otorisasi yang konsisten.
3. Keamanan Jaringan Internal: Komunikasi antar layanan rentan terhadap serangan jika tidak dienkripsi atau diawasi.
4. Pengelolaan Rahasia (Secrets Management): Password, API key, dan token harus disimpan dengan aman.
5. Pengujian Keamanan: Kompleksitas arsitektur menyulitkan proses pengetesan keamanan menyeluruh.
6. Kompleksitas Otentikasi dan Otorisasi: Mengelola hak akses antar layanan dan pengguna menjadi lebih kompleks, terlebih jika menggunakan pendekatan polyglot.
7. Keamanan Komunikasi: Komunikasi antar layanan yang tidak dienkripsi rentan terhadap serangan man-in-the-middle.
8. Deployment Cepat dan Kontinu: CI/CD pipeline yang tidak aman dapat menjadi celah masuk malware atau kode berbahaya ke dalam produksi.

## 3.2 Strategi Keamanan Microservice

1. **Zero Trust Architecture:** Tidak ada layanan yang dipercaya secara default. Setiap permintaan antar layanan harus diautentikasi dan diautorisasi.
2. **API Gateway:** Mengontrol semua lalu lintas masuk dan keluar, serta memberikan fitur seperti rate limiting, logging, dan otentikasi.
3. **Service Mesh (contoh: Istio, Linkerd):** Menyediakan enkripsi lalu lintas antar layanan (mTLS), retry otomatis, serta observabilitas tingkat jaringan.
4. **Identity Federation dan SSO:** Menggunakan protokol OAuth2 dan OpenID Connect dengan IdP seperti Keycloak untuk otentikasi yang konsisten.
5. **Pengelolaan Secrets:** Gunakan tools seperti HashiCorp Vault, AWS Secrets Manager untuk menyimpan rahasia secara terenkripsi.
6. **Logging dan Monitoring Terpusat:** Gunakan ELK Stack, Prometheus, dan Grafana untuk mendeteksi anomali dan insiden keamanan.
7. **Pengetesan Keamanan Berkala:** Lakukan static analysis, penetration test, dan dependency scanning secara rutin.

## 3.3 Studi Kasus dan Praktik Industri

Netflix, sebagai pionir microservice, menerapkan berbagai strategi keamanan mulai dari mutual TLS untuk komunikasi internal hingga penggunaan sistem alert real-time berbasis AI. Uber mengembangkan sistem otorisasi terpusat bernama "Hubble" untuk memastikan semua layanan memiliki kontrol akses berbasis kebijakan yang seragam.

**3.4 Tabel Perbandingan Tools Keamanan Microservice**

|  |  |  |  |
| --- | --- | --- | --- |
| Tool | Fungsi | Kelebihan | Kekurangan |
| Istio | Service mesh & observabilitas | TLS otomatis, tracing, routing cerdas | Kompleks, membutuhkan resource tinggi |
| Keycloak | Identity provider (SSO) | Mendukung OAuth2, OIDC, mudah integrasi | Konfigurasi awal cukup rumit |
| HashiCorp Vault | Secrets management | Aman, audit trail, dynamic secrets | Kurva belajar cukup tinggi |
| Kong / NGINX | API Gateway | Skalabel, fleksibel, banyak plugin | Perlu konfigurasi dan monitoring |
| OPA (Open Policy Agent) | Kebijakan otorisasi | Policy terpusat, integrasi service mesh & API gateway | Perlu pemahaman Rego language |

# BAB IV: KESIMPULAN DAN SARAN

## 4.1 Kesimpulan

Keamanan dalam microservice memerlukan pendekatan yang berbeda dibandingkan dengan sistem monolitik. Kompleksitas dan keterbukaan sistem menuntut adopsi prinsip zero trust, perlindungan komunikasi antar layanan, serta sistem otentikasi dan otorisasi yang kuat. Studi dan praktik terbaik menunjukkan bahwa pendekatan komprehensif yang mencakup API gateway, IdP, dan monitoring mampu meningkatkan postur keamanan secara signifikan.

## 4.2 Saran

1. Terapkan prinsip keamanan sejak tahap perancangan (security by design).
2. Gunakan alat bantu seperti API Gateway dan service mesh untuk meningkatkan kontrol dan visibilitas.
3. Lakukan pelatihan rutin untuk tim developer dan devops tentang praktik keamanan terbaik.
4. Terapkan pipeline CI/CD yang aman dan integrasikan pengetesan keamanan ke dalam proses deployment.
5. Dokumentasikan dan audit semua aktivitas yang berkaitan dengan akses dan perubahan sistem

# DAFTAR PUSTAKA

Di Francesco, P., Lago, P., Malavolta, I. (2021). An Empirical Study of Security Practices for Microservices Systems. arXiv:2112.14927.

Taibi, D., Lenarduzzi, V. (2020). Securing Microservices and Microservice Architectures: A Systematic Mapping Study. arXiv:2003.07262.

Hafiz, M., Shin, Y., Kang, E. (2022). SoK: Security of Microservice Applications: A Practitioners' Perspective on Challenges and Best Practices. arXiv:2202.01612.

Newman, S. (2019). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.

Fowler, M. (2015). Microservices Resource Guide. martinfowler.com.