**Makalah**

**Keamanan dalam Microservices**



|  |  |
| --- | --- |
| Nama | : M.Haikal Habibillah |
| Kelas | : TEKNIK KOMPUTER 2-A |
| NIM | : 2301083004 |

**PROGRAM STUDI TEKNIK KOMPUTER**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNI NEGERI PADANG**

**TAHUN AJARAN 2024/2025**

DAFTAR ISI

1. Pendahuluan
2. Latar Belakang Microservices  
   2.1 Pengertian Microservices  
   2.2 Perbedaan dengan Monolitik  
   2.3 Keuntungan dan Tantangan
3. Risiko dan Ancaman Keamanan  
   3.1 Permukaan Serangan yang Lebar  
   3.2 Komunikasi Antar Layanan  
   3.3 Masalah Autentikasi dan Otorisasi  
   3.4 Pengelolaan Rahasia
4. Strategi Keamanan Microservices  
   4.1 Autentikasi dan Otorisasi  
   4.2 Enkripsi Data  
   4.3 Keamanan API  
   4.4 Manajemen Rahasia  
   4.5 DevSecOps dan Otomatisasi  
   4.6 Segmentasi dan Zero Trust
5. Studi Kasus  
   5.1 Netflix  
   5.2 Amazon
6. Kesimpulan
7. Daftar Pustaka

BAB I

PENDAHULUAN

Perkembangan teknologi informasi yang pesat telah mendorong transformasi besar dalam pengembangan perangkat lunak. Salah satu inovasi arsitektur perangkat lunak yang mendapat banyak perhatian adalah arsitektur microservices. Microservices menawarkan fleksibilitas, skalabilitas, dan kemudahan dalam pengelolaan aplikasi yang kompleks.

Namun, bersamaan dengan keuntungan tersebut, muncul pula tantangan yang signifikan, terutama dalam hal keamanan. Arsitektur ini terdiri dari banyak layanan kecil yang saling berinteraksi melalui jaringan, yang berarti ada lebih banyak titik yang harus diamankan dibandingkan arsitektur monolitik.

Makalah ini akan membahas secara komprehensif tentang keamanan dalam arsitektur microservices, mulai dari latar belakang dan risiko yang ada, hingga strategi keamanan yang dapat diterapkan untuk meminimalisir ancaman. Dengan pemahaman yang baik mengenai aspek keamanan, diharapkan pengembang dan tim keamanan dapat bekerja sama untuk menciptakan sistem yang andal dan aman.

BAB II

LATAR BELAKANG MICROSERVICES

2.1 Pengertian Microservices

Microservices adalah pendekatan arsitektur dalam pengembangan perangkat lunak di mana sebuah aplikasi besar dipecah menjadi layanan-layanan kecil yang berdiri sendiri dan berkomunikasi melalui API. Setiap layanan dalam microservices memiliki tanggung jawab spesifik dan dapat dikembangkan, diuji, dideploy, dan diskalakan secara independen.

2.2 Perbedaan dengan Arsitektur Monolitik

Pada arsitektur monolitik, seluruh fungsi aplikasi digabungkan menjadi satu unit yang besar. Setiap kali ada perubahan atau penambahan fitur, keseluruhan aplikasi harus dibangun dan dideploy ulang. Hal ini berbeda dengan microservices yang memungkinkan pengembangan lebih fleksibel karena setiap layanan dapat diperbarui tanpa memengaruhi layanan lainnya.

Perbandingan singkat:

|  |  |  |
| --- | --- | --- |
| Aspek | Monolitik | Microservices |
| Ukuran kode | Besar dan terintegrasi | Terbagi menjadi layanan kecil |
| Skalabilitas | Sulit diskalakan parsial | Mudah diskalakan per layanan |
| Deploy | Satu kesatuan | Mandiri per layanan |
| Pengembangan tim | Satu tim besar | Tim kecil per layanan |
| Ketahanan kesalahan | Satu error bisa berdampak luas | Error terisolasi pada satu layanan |

2.3 Keuntungan dan Tantangan Microservices

Keuntungan:

* Fleksibilitas Teknologi: Setiap layanan dapat ditulis dalam bahasa pemrograman yang berbeda sesuai kebutuhan.
* Pengembangan Cepat: Tim kecil dapat fokus pada satu layanan saja.
* Skalabilitas Tinggi: Layanan dapat diskalakan secara mandiri.
* Kemandirian Deployment: Pembaruan fitur tidak memengaruhi keseluruhan sistem.

Tantangan:

* Kompleksitas Infrastruktur: Pengelolaan banyak layanan memerlukan orkestrasi.
* Komunikasi Antar Layanan: Rentan terhadap latensi dan gangguan jaringan.
* Monitoring dan Logging: Membutuhkan sistem observabilitas yang canggih.
* Keamanan: Permukaan serangan menjadi lebih luas dan kompleks.

BAB III

RISIKO DAN ANCAMAN KEAMANAN

3.1 Permukaan Serangan yang Luas

Dengan banyaknya layanan yang berjalan secara independen, setiap service endpoint menjadi potensi titik serangan. Semakin banyak layanan yang terekspos ke jaringan, semakin luas pula permukaan serangan (attack surface) yang harus diamankan.

3.2 Komunikasi Antar Layanan yang Rentan

Layanan dalam arsitektur microservices harus saling berkomunikasi, baik secara langsung (synchronous) maupun menggunakan message broker (asynchronous). Tanpa pengamanan seperti enkripsi atau validasi identitas, komunikasi ini dapat disusupi atau dimodifikasi.

3.3 Kelemahan dalam Autentikasi dan Otorisasi

Tanpa penerapan autentikasi dan otorisasi yang konsisten dan terdistribusi, layanan dapat diakses secara tidak sah. Kesalahan konfigurasi atau pemisahan otorisasi yang tidak jelas dapat dimanfaatkan oleh penyerang.

3.4 Masalah Pengelolaan Rahasia dan Konfigurasi

Penggunaan rahasia seperti API key, token, dan password dalam layanan sering kali disimpan secara tidak aman di dalam kode atau file konfigurasi. Hal ini rentan terhadap kebocoran data.

BAB IV

STRATEGI DAN PENDEKATAN KEAMANAN MICROSERVICES

4.1 Autentikasi dan Otorisasi Terdistribusi

Gunakan protokol standar seperti OAuth 2.0 dan OpenID Connect untuk menerapkan autentikasi tunggal (Single Sign-On) dan token-based authentication. JSON Web Token (JWT) menjadi salah satu format yang umum digunakan.

4.2 Penggunaan Enkripsi pada Data dan Komunikasi

Seluruh komunikasi antar layanan harus dienkripsi menggunakan protokol TLS. Selain itu, data sensitif juga harus dienkripsi saat disimpan (at rest) dan saat ditransmisikan (in transit).

4.3 Pengamanan API Endpoint

Gunakan API gateway sebagai perantara untuk mengatur, mengamankan, dan mengawasi lalu lintas ke microservices. API gateway juga dapat digunakan untuk menerapkan rate limiting dan IP filtering.

4.4 Manajemen Rahasia secara Terpusat

Gunakan tools seperti HashiCorp Vault, AWS Secrets Manager, atau Kubernetes Secrets untuk mengelola kredensial, token, dan rahasia lainnya secara aman dan terpusat.

4.5 Integrasi DevSecOps dalam Siklus DevOps

Keamanan harus menjadi bagian integral dalam siklus pengembangan perangkat lunak. Praktik DevSecOps memastikan bahwa keamanan diuji dan diaudit sejak tahap awal pengembangan.

4.6 Penerapan Segmentasi Jaringan dan Zero Trust Architecture

Dengan pendekatan Zero Trust, tidak ada komponen dalam sistem yang dipercaya secara default. Setiap layanan harus memverifikasi identitas layanan lain sebelum berkomunikasi.

4.7 Pemantauan, Logging, dan Respons Insiden

Gunakan sistem centralized logging dan pemantauan real-time (seperti ELK stack, Prometheus, Grafana) untuk mendeteksi anomali dan merespons insiden keamanan dengan cepat.

BAB V

STUDI KASUS PENERAPAN KEAMANAN MICROSERVICES

5.1 Netflix

Netflix adalah salah satu pelopor dalam penggunaan arsitektur microservices. Untuk menjaga keamanan layanan mereka, Netflix menggunakan sistem autentikasi berbasis token, service mesh, dan sistem monitoring terdistribusi yang dapat mendeteksi dan merespons insiden secara cepat.

5.2 Amazon Web Services (AWS)

AWS menyediakan infrastruktur yang aman bagi layanan berbasis microservices dengan berbagai fitur keamanan seperti IAM (Identity and Access Management), VPC (Virtual Private Cloud), dan AWS Shield untuk mitigasi serangan DDoS.

BAB VI

TANTANGAN IMPLEMENTASI KEAMANAN MICROSERVICES DI DUNIA NYATA

Penerapan keamanan microservices di lingkungan nyata menghadapi tantangan seperti:

* Kompleksitas pengelolaan konfigurasi dan rahasia di banyak layanan.
* Kesenjangan skill antara tim pengembang dan tim keamanan.
* Koordinasi lintas tim dalam menerapkan kebijakan keamanan secara konsisten.
* Kebutuhan akan observabilitas dan automasi dalam deteksi serta mitigasi ancaman.

BAB VII - KESIMPULAN

Keamanan dalam arsitektur microservices merupakan aspek penting yang tidak bisa diabaikan. Sistem yang terdiri dari banyak layanan mandiri memiliki permukaan serangan yang lebih luas dan memerlukan pendekatan keamanan yang terdistribusi dan konsisten. Praktik terbaik seperti enkripsi komunikasi, autentikasi terstandarisasi, pengamanan API, serta integrasi DevSecOps dapat membantu menjaga sistem tetap aman.

Melalui studi kasus perusahaan besar seperti Netflix dan AWS, kita dapat melihat bahwa dengan perencanaan dan teknologi yang tepat, tantangan keamanan microservices dapat diatasi dengan efektif.

DAFTAR PUSTAKA

* Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.
* Fowler, M. (2014). Microservices: a definition of this new architectural term. martinfowler.com.
* Amazon Web Services. (2024). Microservices on AWS. aws.amazon.com.
* NIST Special Publication 800-207. (2020). Zero Trust Architecture.
* Netflix Technology Blog. techblog.netflix.com.