# MAKALAH

# KEAMANAN DALAM MICROSERVICES

## PEMROGRAMAN MICROSERVICES TEORI

**Dosen Pengampu :**

Ervan Asri, S.Kom., M.Kom


**Disusun Oleh:**

**FATHURRAHMAN AL HAFIZ**

**2201082005**

**SEMESTER 4**

**PROGRAM STUDI D3 TEKNIK KOMPUTER**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI PADANG**

**2024/2025**

# DAFTAR ISI

# BAB I
# PENDAHULUAN

## 1.1. Latar Belakang

Dalam era digital yang berkembang pesat, arsitektur microservices telah menjadi pendekatan populer dalam pengembangan perangkat lunak. Microservices memungkinkan aplikasi dipecah menjadi layanan-layanan kecil yang independen, sehingga memudahkan pengembangan, pengujian, dan deployment secara terpisah. Pendekatan ini menawarkan fleksibilitas dan skalabilitas yang tinggi, memungkinkan setiap layanan dikembangkan dengan teknologi yang paling sesuai dengan kebutuhannya. Namun, di balik berbagai keunggulannya, arsitektur microservices juga menghadirkan tantangan baru, terutama dalam aspek keamanan. Kompleksitas yang meningkat akibat banyaknya layanan yang berkomunikasi melalui jaringan memperluas permukaan serangan dan meningkatkan risiko terhadap ancaman keamanan. Oleh karena itu, penting untuk memahami dan mengimplementasikan strategi keamanan yang efektif dalam arsitektur microservices guna melindungi data dan memastikan integritas sistem.

## 1.2. Rumusan Masalah

Adapun rumusan masalah sebagai berikut:

1. Apa saja ancaman keamanan yang umum terjadi dalam arsitektur microservices?
2. Bagaimana strategi dan teknik yang efektif untuk mengamankan layanan-layanan dalam microservices?
3. Bagaimana implementasi praktik terbaik dalam menjaga keamanan microservices?

## 1.3. Tujuan Penulisan

Penulisan ini bertujuan untuk:

1. Mengidentifikasi dan memahami berbagai ancaman keamanan yang dihadapi dalam arsitektur microservices.
2. Menganalisis strategi dan teknik yang dapat diterapkan untuk meningkatkan keamanan dalam microservices.

3. Memberikan rekomendasi praktik terbaik dalam implementasi keamanan microservices.

## 1.4. Manfaat Penulisan

Adapun Manfaat masalah sebagai berikut:

1. Memberikan wawasan kepada pengembang perangkat lunak mengenai pentingnya aspek keamanan dalam arsitektur microservices.
2. Menyediakan panduan bagi organisasi dalam mengimplementasikan langkah-langkah keamanan yang efektif untuk melindungi sistem berbasis microservices.
3. Meningkatkan kesadaran akan potensi risiko dan ancaman yang mungkin timbul serta cara mitigasinya dalam lingkungan microservices.

# BAB II
# KONSEP DASAR MICROSERVICES

## 2.1. Pengertian Microservices

Arsitektur microservices adalah pendekatan dalam pengembangan perangkat lunak yang membagi aplikasi menjadi layanan-layanan kecil yang independen dan terdistribusi. Setiap layanan memiliki fungsi spesifik dan beroperasi secara mandiri, berkomunikasi melalui antarmuka yang terdefinisi dengan baik, seperti API. Pendekatan ini berbeda dengan arsitektur monolitik, di mana semua komponen aplikasi digabungkan menjadi satu kesatuan.

## 2.2. Karakteristik Arsitektur Microservices

Beberapa karakteristik utama dari arsitektur microservices meliputi:

1. Modularitas: Setiap layanan merupakan modul independen yang menangani satu fungsi bisnis tertentu, memungkinkan pengembangan dan pemeliharaan yang lebih mudah.
2. Independensi Implementasi: Setiap layanan dapat dikembangkan, diuji, dan di-deploy secara terpisah tanpa mempengaruhi layanan lain, memberikan fleksibilitas dalam pengembangan.
3. Fleksibilitas Teknologi: Pengembang dapat menggunakan bahasa pemrograman, database, dan teknologi yang berbeda untuk setiap layanan sesuai kebutuhan spesifik.
4. Skalabilitas: Setiap layanan dapat diskalakan secara independen, memungkinkan penyesuaian kapasitas berdasarkan permintaan tanpa harus menskalakan seluruh aplikasi.
5. Toleransi terhadap Kegagalan: Kegagalan pada satu layanan tidak langsung mempengaruhi layanan lain, sehingga meningkatkan ketahanan sistem secara keseluruhan.

## 2.3. Perbandingan Microservices dan Monolith

Perbedaan utama antara arsitektur microservices dan monolitik dapat dilihat pada tabel berikut:

| Aspek | Arsitektur Monolitik | Arsitektur Microservices |
|---|---|---|
| Skalabilitas | Sulit diskalakan; peningkatan sumber daya memerlukan deployment ulang seluruh aplikasi. | Mudah diskalakan; layanan dapat diskalakan secara individu sesuai kebutuhan. |
| Pengembangan | Tim harus berkoordinasi untuk semua perubahan kode, memperlambat proses pengembangan. | Tim dapat bekerja secara independen pada layanan yang berbeda, mempercepat pengembangan. |
| Deployment | Setiap perubahan memerlukan deployment ulang seluruh aplikasi. | Layanan dapat di-deploy dan diperbarui secara terpisah tanpa mempengaruhi sistem lain. |
| Ketahanan | Kegagalan satu komponen dapat menyebabkan kegagalan seluruh aplikasi. | Kegagalan satu layanan tidak mempengaruhi layanan lainnya, meningkatkan ketahanan sistem. |

## 2.4. Keuntungan dan Tantangan Microservices

Keuntungan:

1. Skalabilitas yang Lebih Baik**:** Layanan dapat diskalakan secara independen sesuai kebutuhan, memungkinkan efisiensi sumber daya.
2. Fleksibilitas dalam Pengembangan**:** Setiap layanan dapat dikembangkan dengan teknologi yang paling sesuai, memberikan kebebasan dalam pemilihan alat dan bahasa pemrograman.

3. Peningkatan Ketahanan Sistem**:** Kegagalan pada satu layanan tidak langsung mempengaruhi layanan lain, sehingga sistem lebih tahan terhadap gangguan**.**
4. Pengembangan dan Deployment yang Cepat: Tim dapat bekerja secara paralel pada layanan yang berbeda, memungkinkan siklus pengembangan yang lebih cepat dan deployment yang lebih sering.

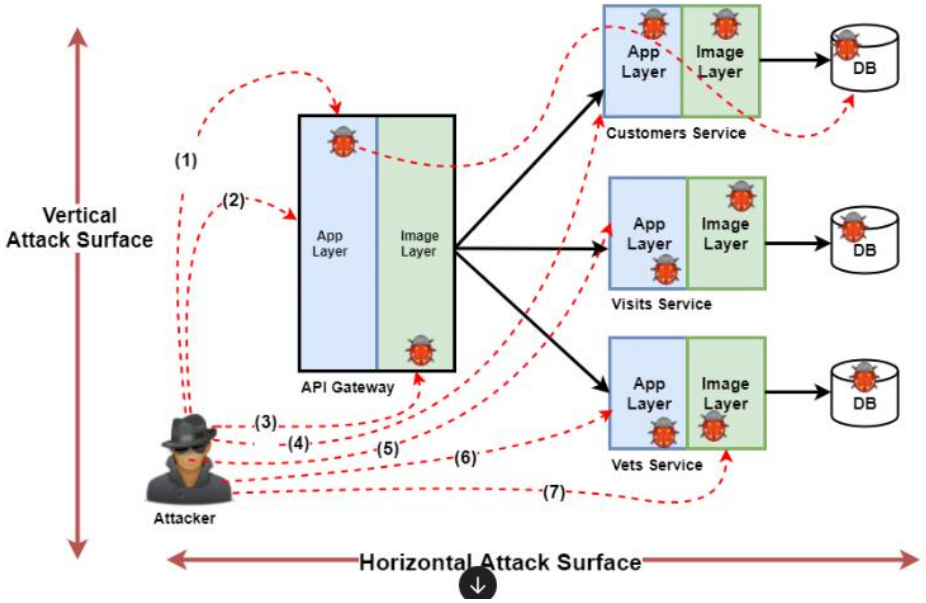
Tantangan:

1. Kompleksitas Manajemen: Mengelola banyak layanan independen memerlukan koordinasi yang baik dan alat manajemen yang efektif.
2. Komunikasi Antar Layanan: Memastikan komunikasi yang efisien dan aman antara layanan-layanan memerlukan desain dan protokol yang tepat.
3. Pengujian yang Rumit: Pengujian integrasi menjadi lebih kompleks karena melibatkan banyak layanan yang saling berinteraksi.
4. Keamanan: Dengan banyaknya titik komunikasi, potensi risiko keamanan meningkat dan memerlukan perhatian khusus dalam desain dan implementasi.

# BAB III
# ANCAMAN KEAMANAN DALAM MICROSERVICES

## 3.1. Permukaan Serangan yang Luas

Dalam arsitektur microservices, setiap layanan beroperasi secara independen dan berkomunikasi melalui jaringan, yang secara signifikan memperluas permukaan serangan. Setiap endpoint layanan menjadi titik potensial bagi penyerang untuk mengeksploitasi kerentanan. Analisis permukaan serangan membantu mengidentifikasi dan memitigasi risiko ini.




*Gambar 1. 1 horizontal attack surface*

Gambar di atas menunjukkan permukaan serangan horizontal dan vertikal dalam aplikasi microservices, mengilustrasikan berbagai titik yang dapat dieksploitasi oleh penyerang.

## 3.2. Ancaman pada API dan Komunikasi Layanan

Mengelola autentikasi dan otorisasi di lingkungan microservices menantang karena banyaknya layanan yang harus diverifikasi. Penggunaan standar seperti OAuth 2.0 dan OpenID Connect dapat membantu memastikan bahwa hanya entitas yang berwenang yang dapat mengakses layanan tertentu.

### 3.3. Risiko dari Service Discovery dan Config Management

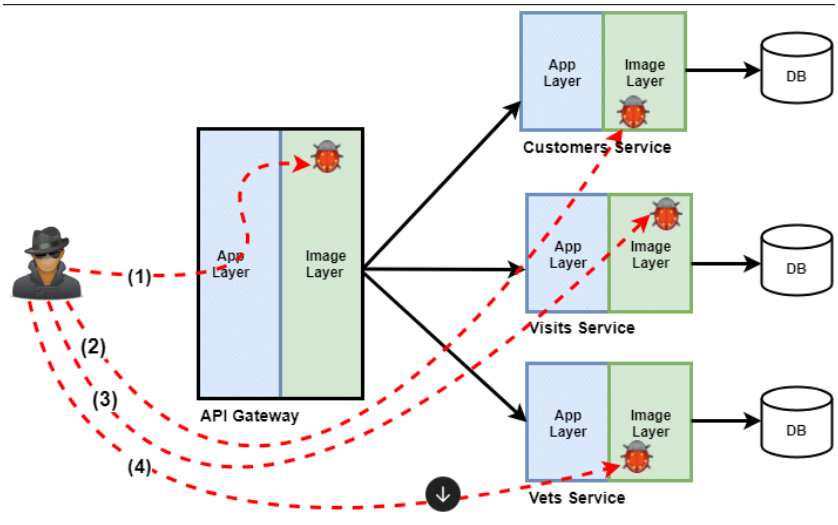
Service discovery dan manajemen konfigurasi memungkinkan layanan menemukan dan berkomunikasi satu sama lain secara dinamis. Namun, jika mekanisme ini tidak diamankan, penyerang dapat mengeksploitasi untuk mengarahkan layanan ke endpoint berbahaya atau mengakses informasi sensitif.

### 3.4. Ancaman dari Dependencies Pihak Ketiga

Microservices sering memanfaatkan pustaka dan layanan pihak ketiga. Jika komponen ini memiliki kerentanan, mereka dapat menjadi pintu masuk bagi penyerang. Penting untuk secara rutin memantau dan memperbarui dependencies serta memastikan bahwa mereka berasal dari sumber yang tepercaya.

### 3.5. Serangan Internal dan Insider Threats

Ancaman tidak selalu berasal dari luar; individu dalam organisasi dengan akses yang sah dapat menyalahgunakan hak mereka. Menerapkan prinsip hak akses minimal dan memantau aktivitas internal dapat membantu mendeteksi dan mencegah ancaman dari dalam.




*Gambar 1. 2 model ancaman internal*

Gambar ini menggambarkan model ancaman internal dalam sistem jaringan, menyoroti bagaimana individu dengan akses sah dapat menyalahgunakan hak mereka.

# BAB IV
# STRATEGI DAN TEKNIK KEAMANAN MICROSERVICES

## 4.1. Secure API Gateway

API Gateway berfungsi sebagai pintu gerbang utama yang mengelola lalu lintas antara klien dan layanan microservices. Dengan mengamankan API Gateway, Anda dapat mengimplementasikan autentikasi dan otorisasi terpusat, pembatasan laju (rate limiting), serta pemantauan dan pencatatan aktivitas untuk mendeteksi dan merespons ancaman dengan cepat.

## 4.2. Penggunaan Protokol Aman (HTTPS, mTLS)

1. HTTPS: Mengamankan data dalam transmisi antara klien dan layanan dengan mengenkripsi komunikasi, mencegah serangan seperti man-in-the-middle.
2. mTLS (Mutual TLS): Memperluas TLS standar dengan memverifikasi identitas kedua belah pihak dalam komunikasi layanan-ke-layanan, memastikan bahwa hanya layanan yang sah yang berkomunikasi.

## 4.3. OAuth2 dan OpenID Connect untuk Autentikasi

1. OAuth2: Kerangka kerja otorisasi yang memungkinkan aplikasi pihak ketiga mendapatkan akses terbatas ke sumber daya pengguna tanpa membagikan kredensial.
2. OpenID Connect: Lapisan identitas yang dibangun di atas OAuth2, memungkinkan verifikasi identitas pengguna dan memperoleh informasi profil dasar.

## 4.4. Role-Based Access Control (RBAC) dan Attribute-Based Access Control (ABAC)

RBAC Memberikan akses berdasarkan peran pengguna dalam organisasi, menyederhanakan manajemen izin dengan mengelompokkan pengguna berdasarkan fungsi pekerjaan mereka. Sedangkan ABAC Menggunakan atribut

seperti departemen, lokasi, atau waktu untuk membuat keputusan akses, memungkinkan kontrol yang lebih granular dibandingkan RBAC.

### 4.5. Isolasi Layanan dan Keamanan Kontainer

Isolasi Layanan Menjalankan setiap layanan dalam lingkungan terisolasi untuk mencegah satu layanan yang terkompromi mempengaruhi yang lain. Dan Keamanan Kontainer Mengamankan image kontainer, membatasi hak istimewa, dan memantau aktivitas untuk mendeteksi perilaku mencurigakan.

### 4.6. Logging, Monitoring, dan Auditing

1. Logging: Mencatat semua aktivitas dan transaksi untuk analisis forensik dan deteksi anomali.
2. Monitoring: Memantau kinerja dan kesehatan layanan secara real-time untuk mendeteksi dan merespons insiden dengan cepat.
3. Auditing: Meninjau log dan aktivitas secara berkala untuk memastikan kepatuhan terhadap kebijakan keamanan dan regulasi.

### 4.7. Service Mesh dan Zero Trust Security

Lapisan infrastruktur (Service Mesh) yang menangani komunikasi layanan-ke-layanan, menyediakan fitur seperti load balancing, pemantauan, dan keamanan. dan Pendekatan keamanan (Zero Trust Security) yang tidak mempercayai apa pun secara default, memverifikasi setiap permintaan seolah-olah berasal dari jaringan yang tidak aman, sering diimplementasikan melalui service mesh dengan mTLS dan kebijakan akses ketat.

### 4.8. Dependency Scanning dan Supply Chain Security

1. Dependency Scanning: Memeriksa pustaka dan komponen pihak ketiga untuk kerentanan yang diketahui, memastikan bahwa aplikasi tidak menggunakan komponen yang rentan.
2. Supply Chain Security: Mengamankan seluruh rantai pasokan perangkat lunak, dari pengembangan hingga distribusi, untuk mencegah penyusupan kode berbahaya atau komponen yang tidak sah.

# BAB VI
# PENUTUP

## 6.1. Kesimpulan

Dalam era transformasi digital, arsitektur microservices telah menjadi pilihan utama dalam pengembangan perangkat lunak karena fleksibilitas dan skalabilitasnya. Namun, pendekatan ini juga membawa tantangan keamanan yang signifikan, termasuk perluasan permukaan serangan, kompleksitas dalam autentikasi dan otorisasi, serta risiko dari dependensi pihak ketiga. Untuk mengatasi tantangan ini, penerapan strategi keamanan seperti penggunaan API Gateway yang aman, protokol komunikasi terenkripsi (HTTPS dan mTLS), serta implementasi mekanisme autentikasi dan otorisasi yang kuat seperti OAuth2 dan OpenID Connect menjadi krusial. Selain itu, pendekatan seperti Role-Based Access Control (RBAC) dan Attribute-Based Access Control (ABAC), isolasi layanan melalui keamanan kontainer, serta penerapan Service Mesh dengan prinsip Zero Trust Security, dapat meningkatkan posture keamanan secara keseluruhan. Penerapan praktik terbaik ini memungkinkan organisasi untuk membangun arsitektur microservices yang tidak hanya efisien dan scalable, tetapi juga aman dan andal

## 6.2. Saran

Untuk memperkuat keamanan dalam arsitektur microservices, disarankan agar organisasi:

1. Mengintegrasikan Keamanan Sejak Awal Pengembangan: Menerapkan pendekatan "secure by design" dengan melibatkan tim keamanan dalam setiap tahap siklus hidup pengembangan perangkat lunak, memastikan bahwa aspek keamanan dipertimbangkan sejak awal.
2. Melakukan Pelatihan Keamanan Berkala untuk Tim: Memberikan edukasi dan pelatihan rutin kepada tim pengembang dan operasional mengenai ancaman keamanan terbaru dan praktik terbaik dalam mengamankan microservices.
3. Dapat menjadikan sebagai pusat penelitian mengenai dampak gempa dan strategi mitigasi bencana, sekaligus menjadi tempat untuk mengedukasi generasi muda mengenai pentingnya kesiapsiagaan menghadapi bencana.

4. Menggunakan Alat Pemantauan dan Logging yang Efektif:
   Mengimplementasikan sistem pemantauan dan logging yang komprehensif untuk mendeteksi aktivitas mencurigakan dan merespons insiden keamanan secara cepat dan tepat.

## DAFTAR PUSTAKA


*Behrens, S. (2017, 15 Agustus). How Netflix DDoS'd Itself To Help Protect the Entire Internet. Wired. Diakses pada 8 April 2025, dari https://www.wired.com/story/netflix-ddos-attack*

*OWASP Foundation. (n.d.). Microservices Security - OWASP Cheat Sheet Series. Diakses pada 8 April 2025, dari https://cheatsheetseries.owasp.org/cheatsheets/Microservices_Security_Cheat_Sheet.html*

*Solo.io. (n.d.). How service mesh supports a zero trust architecture. Diakses pada 8 April 2025, dari https://www.solo.io/blog/service-mesh-zero-trust*

*Red Hat. (n.d.). Istio Security: Running Microservices on Zero-Trust Networks. Diakses pada 8 April 2025, dari https://www.redhat.com/en/blog/istio-security-running-microservices-on-zero-trust-networks*

*Tetrate.io. (n.d.). Zero Trust Network for Microservices. Diakses pada 8 April 2025, dari https://tetrate.io/blog/zero-trust-network-for-microservices*

*Kanjilal, J. (2021, 15 Maret). 8 fundamental microservices security best practices. TechTarget. Diakses pada 8 April 2025, dari https://www.techtarget.com/searchapparchitecture/tip/4-fundamental-microservices-security-best-practices*

*The New Stack. (n.d.). Zero Trust Security with Service Mesh. Diakses pada 8 April 2025, dari https://thenewstack.io/zero-trust-security-with-service-mesh*

*Snyk. (n.d.). Microservices security: 6 Best practice tips. Diakses pada 8 April 2025, dari https://snyk.io/blog/microservices-security*

*Tigera. (n.d.). Top 10 Microservices Security Patterns. Diakses pada 8 April 2025, dari https://www.tigera.io/learn/guides/microservices-security*

*Alibaba Cloud. (n.d.). Zero Trust Security with Cloud-Native Microservices and Containers. Medium. Diakses pada 8 April 2025, dari https://alibaba-cloud.medium.com/zero-trust-security-part-3-zero-trust-security-with-cloud-native-microservices-and-containers-75961aa2e65e*

*Imesh.ai. (n.d.). Zero Trust Network for Microservices with Istio. Diakses pada 8 April 2025, dari https://imesh.ai/blog/zero-trust-network-for-microservices-with-istio*

*DevOps.com. (n.d.). How Service Mesh Enables a Zero-Trust Network. Diakses pada 8 April 2025, dari https://devops.com/how-service-mesh-enables-a-zero-trust-network*

*Kong Inc. (n.d.). Guide to Zero Trust Security and Microservices Adoption. Diakses pada 8 April 2025, dari https://konghq.com/blog/enterprise/the-importance-of-zero-trust-security-when-making-the-microservices-move*

*Palladino, M. (2021, 10 Februari). How to Implement Zero Trust Security with Service Mesh. Kong Inc. Diakses pada 8 April 2025, dari https://konghq.com/blog/engineering/zero-trust-service-mesh-security*