Microservice Security

Jingyue Li

31 March 2025

Agenda

- Microservice architecture
- Microservice security challenges
- Microservice security countermeasures

Monolithic Architecture

- Limited scalability
- Sigle-point of failure
- Need to rebuild an entire development to change a small constraint or check.

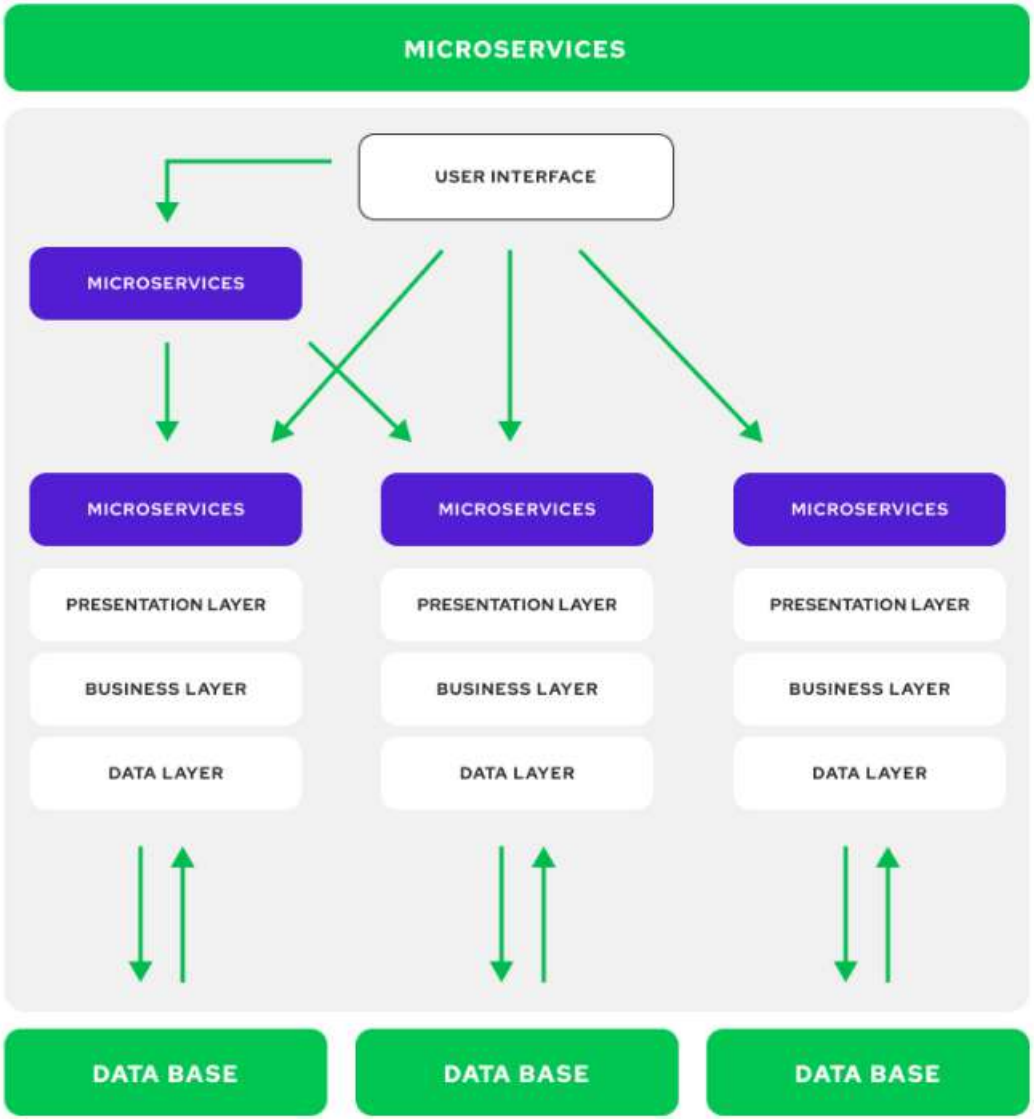
*

* https://alokai.com/blog/microservices-examples

Microservice Architecture

- Loosely coupled and communicates via APIs
- Highly maintainable and testable
- Independently deployable
- Organized around business capabilities

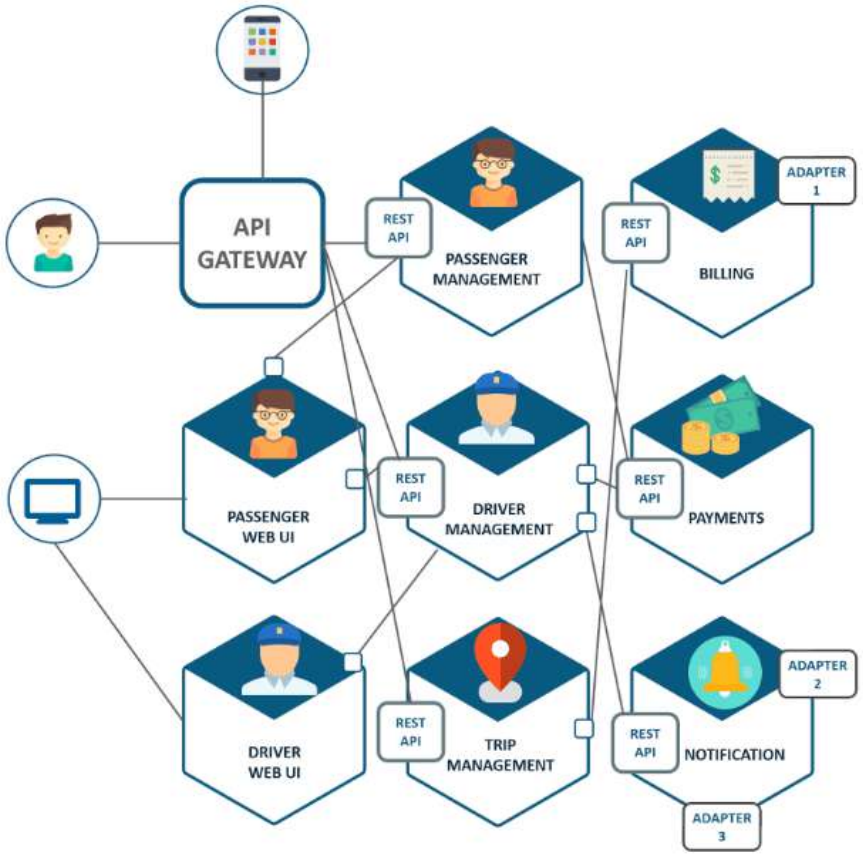
*



* https://alokai.com/blog/microservices-examples

Examples of Microservices in Action*

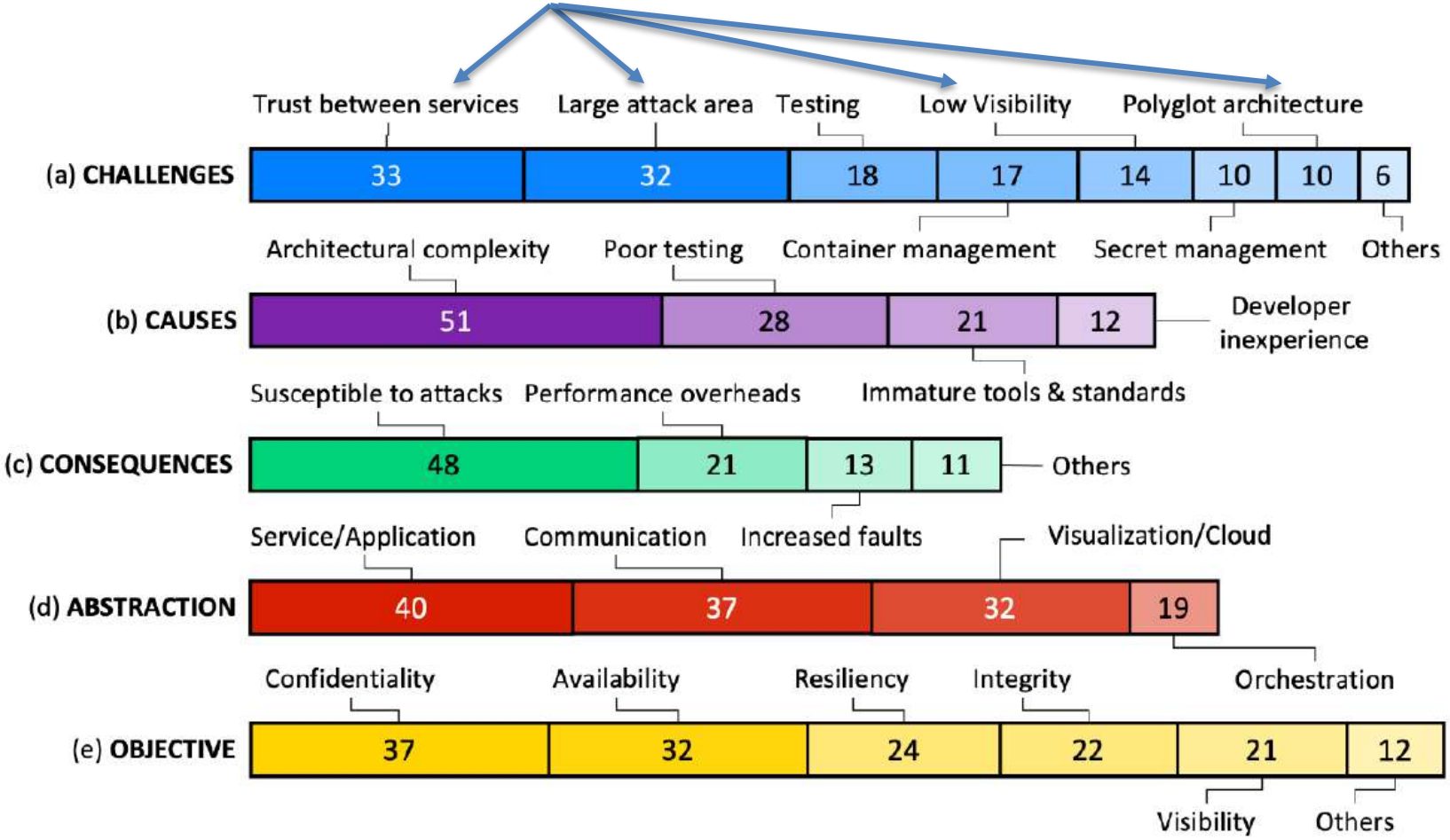
- Uber
- Amazon (Amazon AWS and Apollo)
- Nexflix architecture consisted of over 700 loosely coupled microservices (by 2017)



Uber's microservices architecture from Dzone

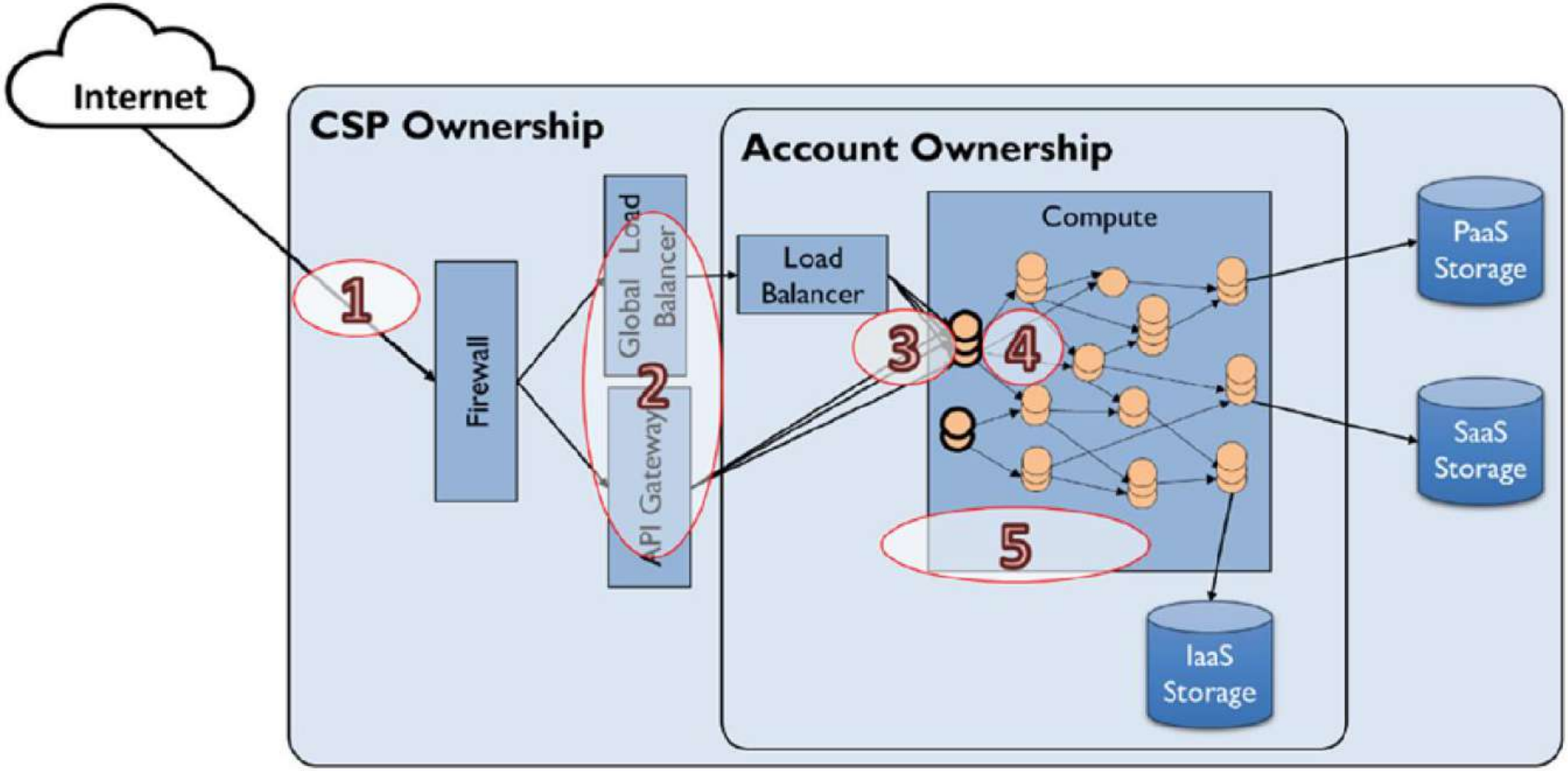
* https://blog.dreamfactory.com/microservices-examples/

Microservice security challenges



* Billawa et al. SoK: Security of Microservice Applications: A Practitioners' Perspective on Challenges and Best Practices (ARES '22).

Large attack area



Security hotspots in microservices cloud deployment*

* Chapter 9 of the book “Cloud-Based Microservices: Techniques, Challenges, and Solutions” by Chandra Rajasekharaiah.

Trust between services

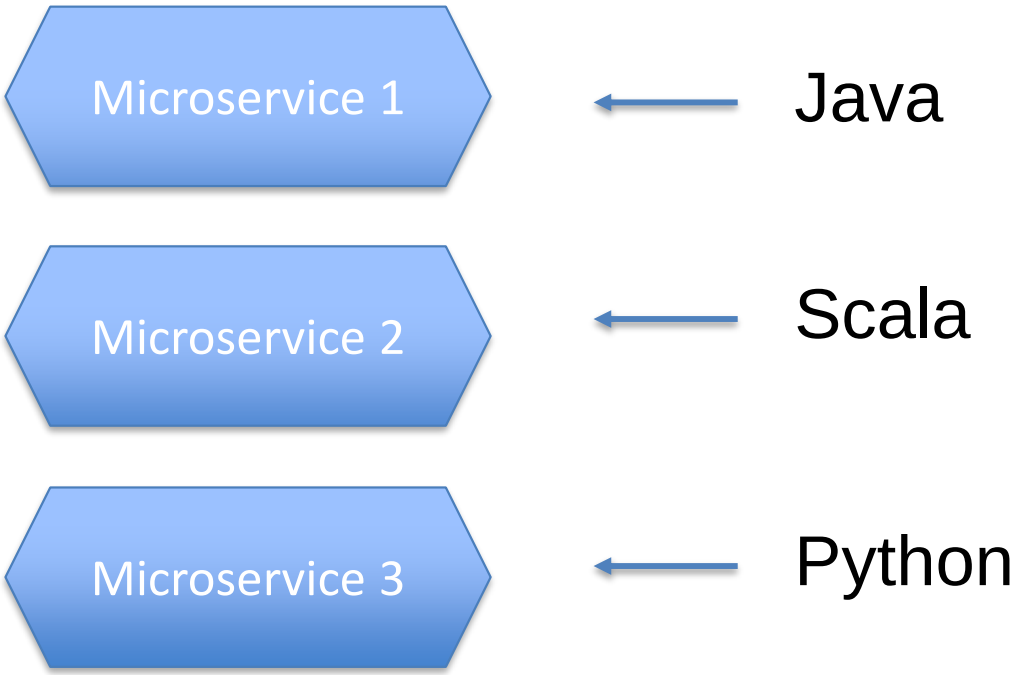
- Some services deployed on the cloud might be malicious.
- Communication between the services could be insecure
  - Insufficient authentication
  - Improper authorization
- Malicious microservices can compromise other services they communicate with.

Low visibility

- Microservice architecture applications are usually deployed on the cloud.
- Unlike an infrastructure entirely owned and managed by enterprises, cloud infrastructure tends to be opaque and disparate.
- We encounter challenges
  - Securing Internet-facing service endpoints
  - Federating access management from enterprise to cloud
  - Securing inter-service communication on an opaque infrastructure

Polyglot architecture

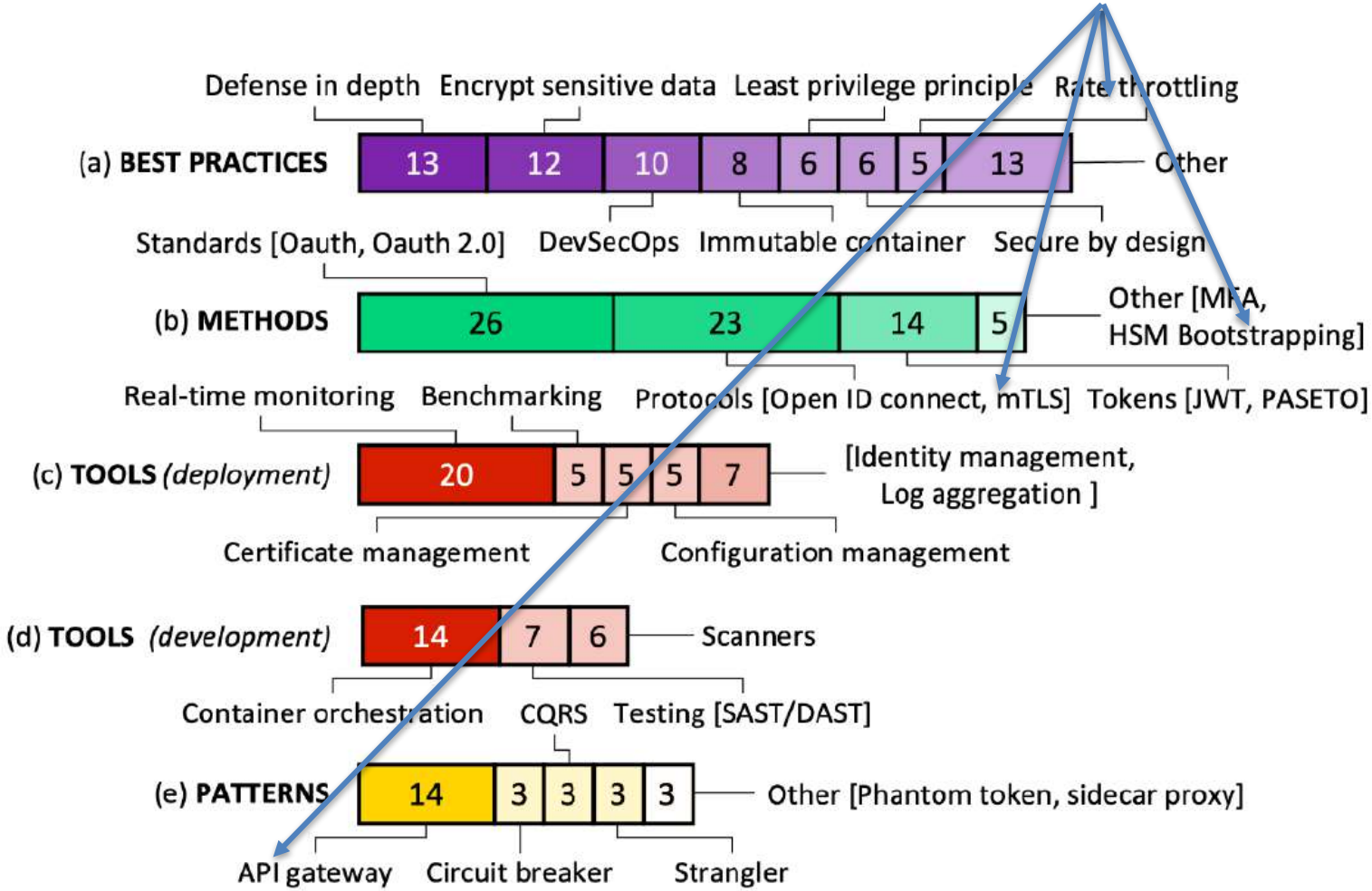
- Polyglot: knowing or using several languages
  - Polu: Greek for many
  - Glotta: Greek for tongue or language



Polyglot architecture security issue

- Different programming languages have different life cycles and versions
- Need the right security expertise at every framework in the stack (along with their particular issues)

Microservice security countermeasures



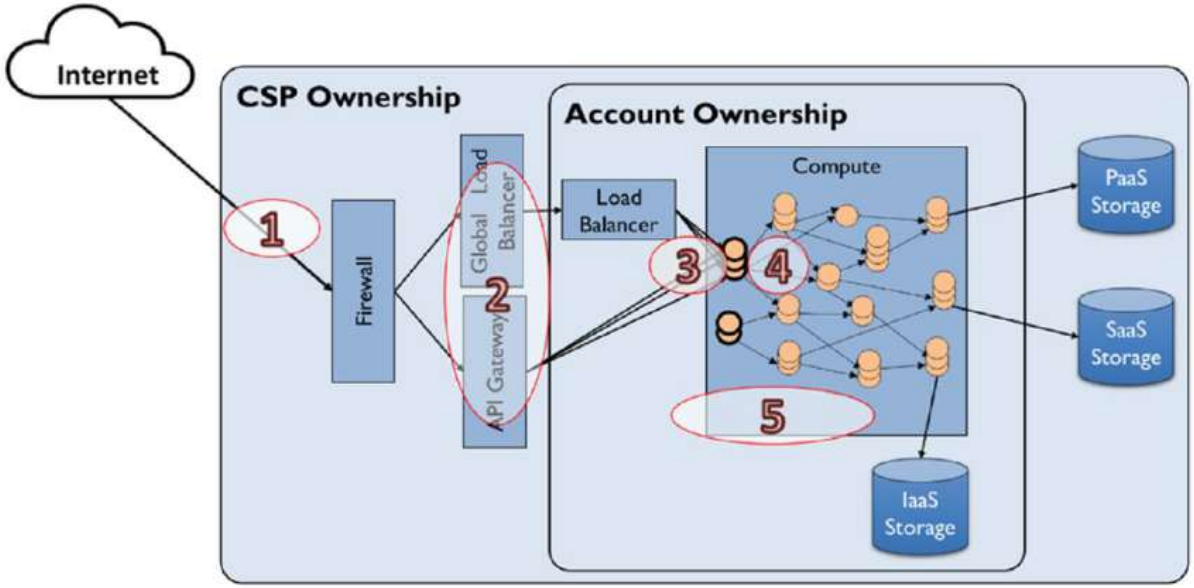
* Billawa et al. SoK: Security of Microservice Applications: A Practitioners' Perspective on Challenges and Best Practices (ARES '22).

Rate throttling

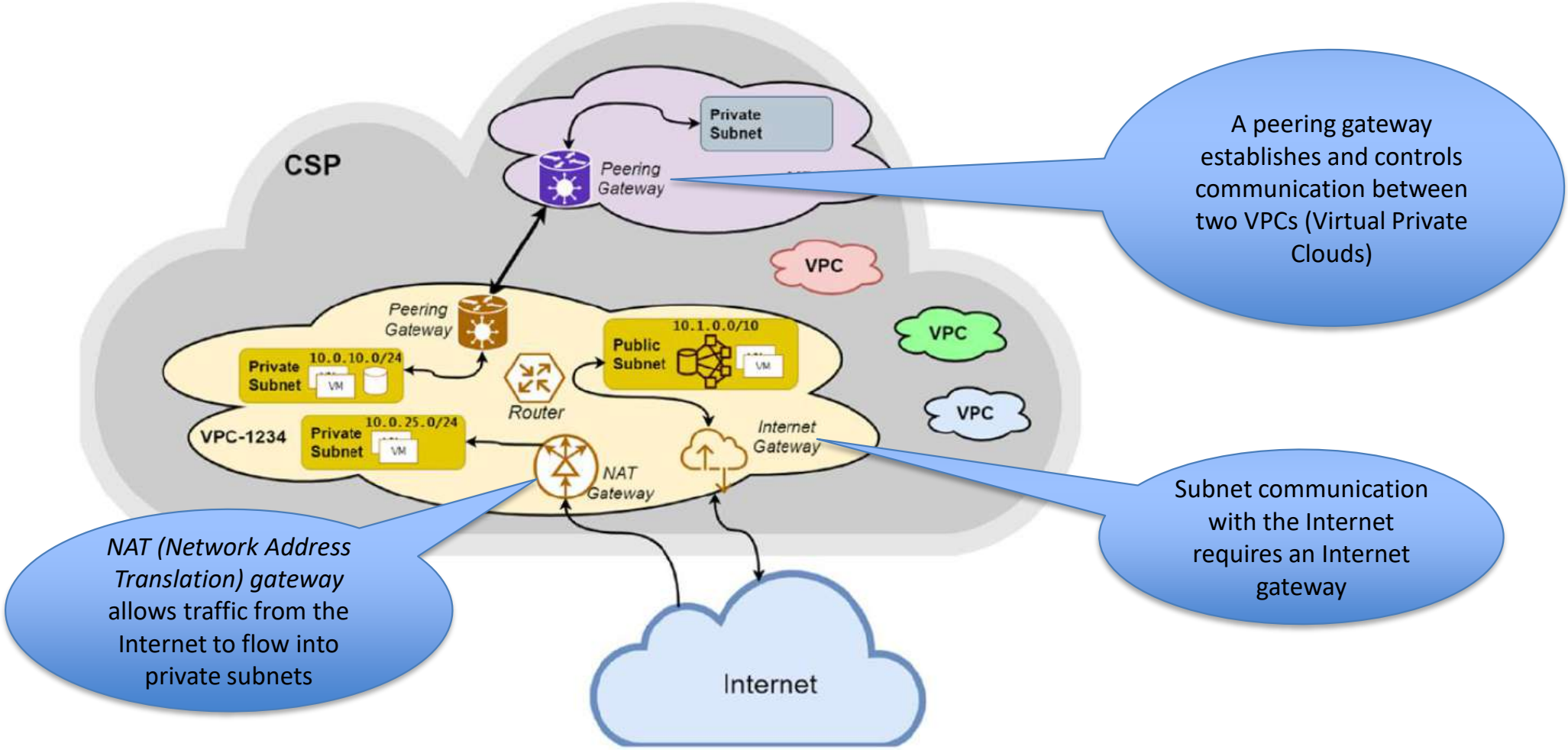
- To defend against DoS attacks
- Microservices architecture-based applications expose hundreds, if not thousands, of API endpoints for external use
- Throttle traffic flow based on configuration
  - Identify that the congestion is approaching
  - Send the feedback on time to the senders that are creating congestion and warn them not to send more packets in an already congested network

Authentication and authorization

- At API-gateway
- From API gateway to microservices
- Between microservices
- At microservices



At API-gateway



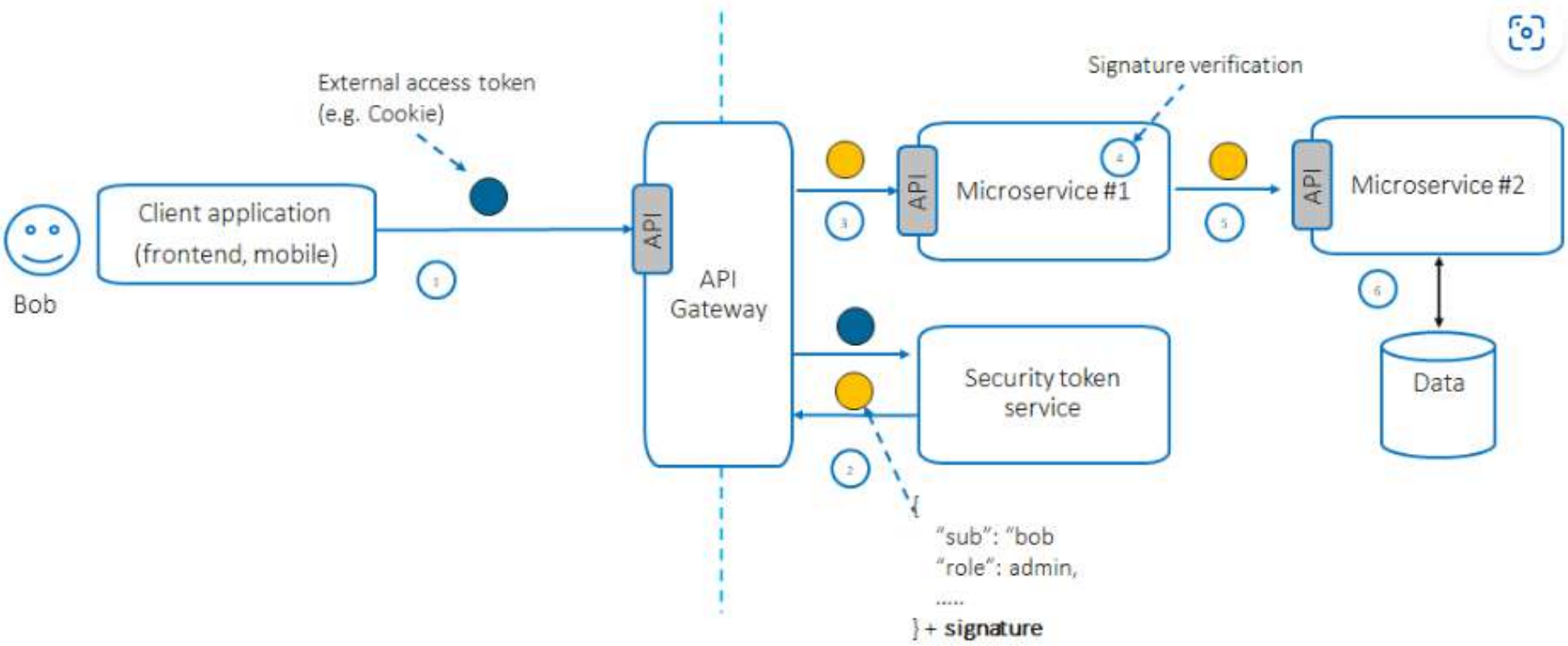
Virtual Private Clouds on a Cloud Service Provider (CSP) *

* Chapter 9 of the book “Cloud-Based Microservices: Techniques, Challenges, and Solutions” by Chandra Rajasekharaiah

API-gateway security

- Enforce verifiable client identification at entry points
  - E.g., Mandate every request to contain a client-ID or access token
- Controlling access by providing authorization policies
  - E.g., Who (person and other microservices) can access what
- Throttling request traffic and thus providing defense against DoS attacks
  - E.g., limit usage (maximum number of requests per time unit, the largest number of simultaneous requests allowed, etc.)

From API to microservice: External Entity Identity Propagation



* https://cheatsheetseries.owasp.org/cheatsheets/Microservices_Security_Cheat_Sheet.html

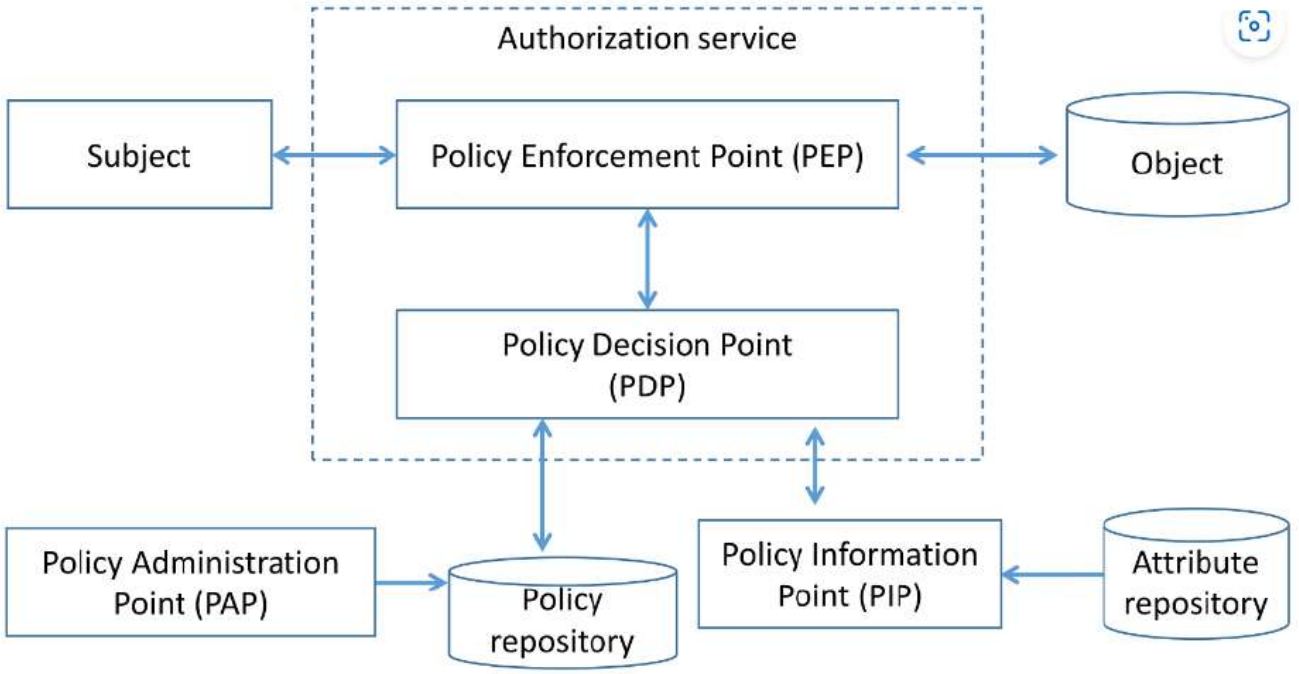
Between Microservices

- Mutual transport layer security (mTLS)
  - Each microservice in the deployment has to carry a public/private key pair and use that key pair to authenticate to the recipient microservices via mTLS.

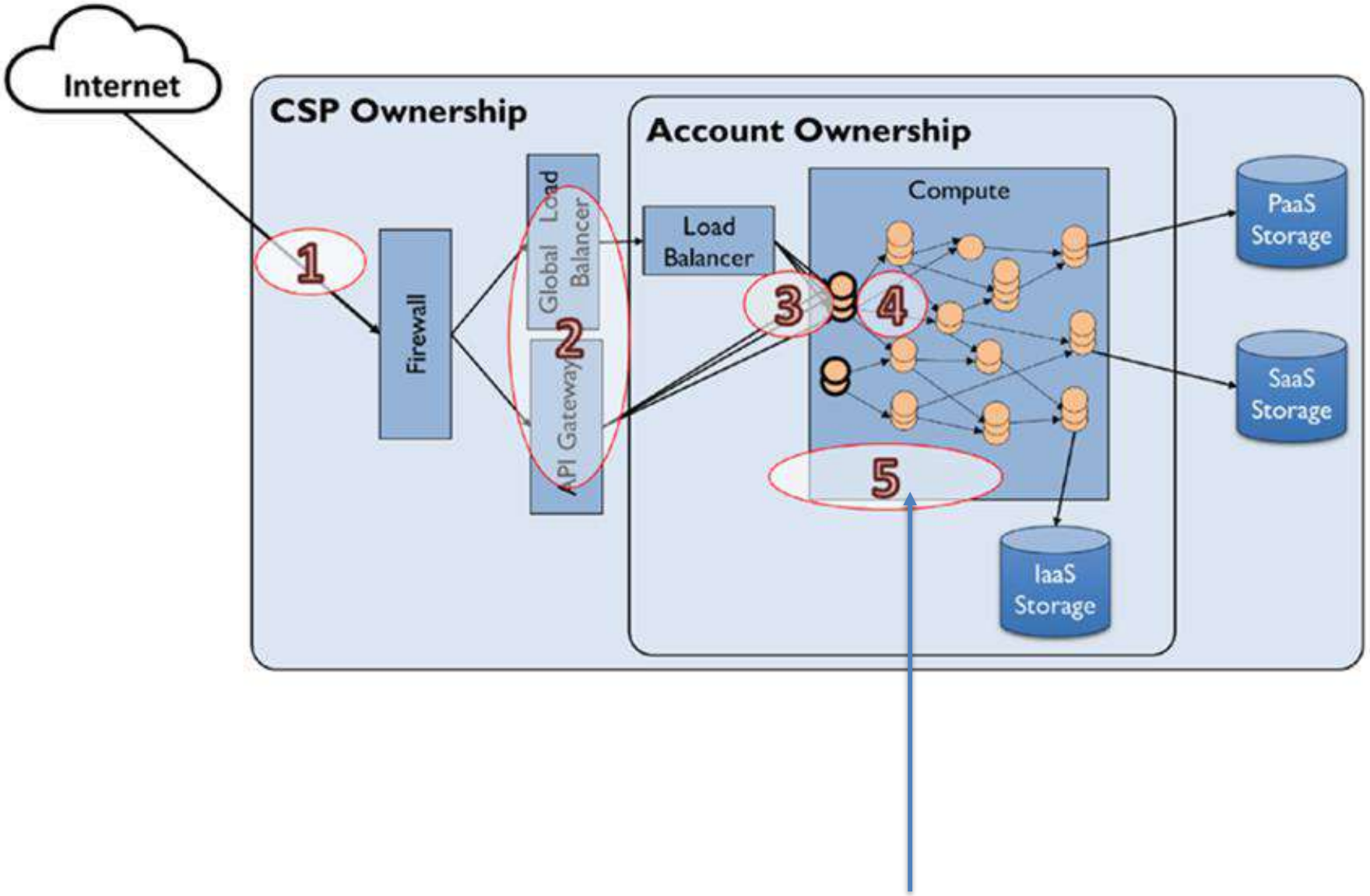
- Token-based
  - The caller microservice can obtain a signed token by invoking a special security token service using its own service ID and password and then attaching it to every outgoing request.

At Service: Service-level authorization

- Gives each microservice more control to enforce access control policies



* https://cheatsheetseries.owasp.org/cheatsheets/Microservices_Security_Cheat_Sheet.html


Internet
CSP Ownership
Account Ownership
Firewall
Global Load Balancer
API Gateway
Load Balancer
Compute
PaaS Storage
SaaS Storage
IaaS Storage
1
2
3
4
5

Trusted container and binaries


The Banyan Security Blog

Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities

by Tarun Desikan | May 05, 2015

https://www.banyansecurity.io/blog/over-30-of-official-images-in-docker-hub-contain-high-priority-security-vulnerabilities/

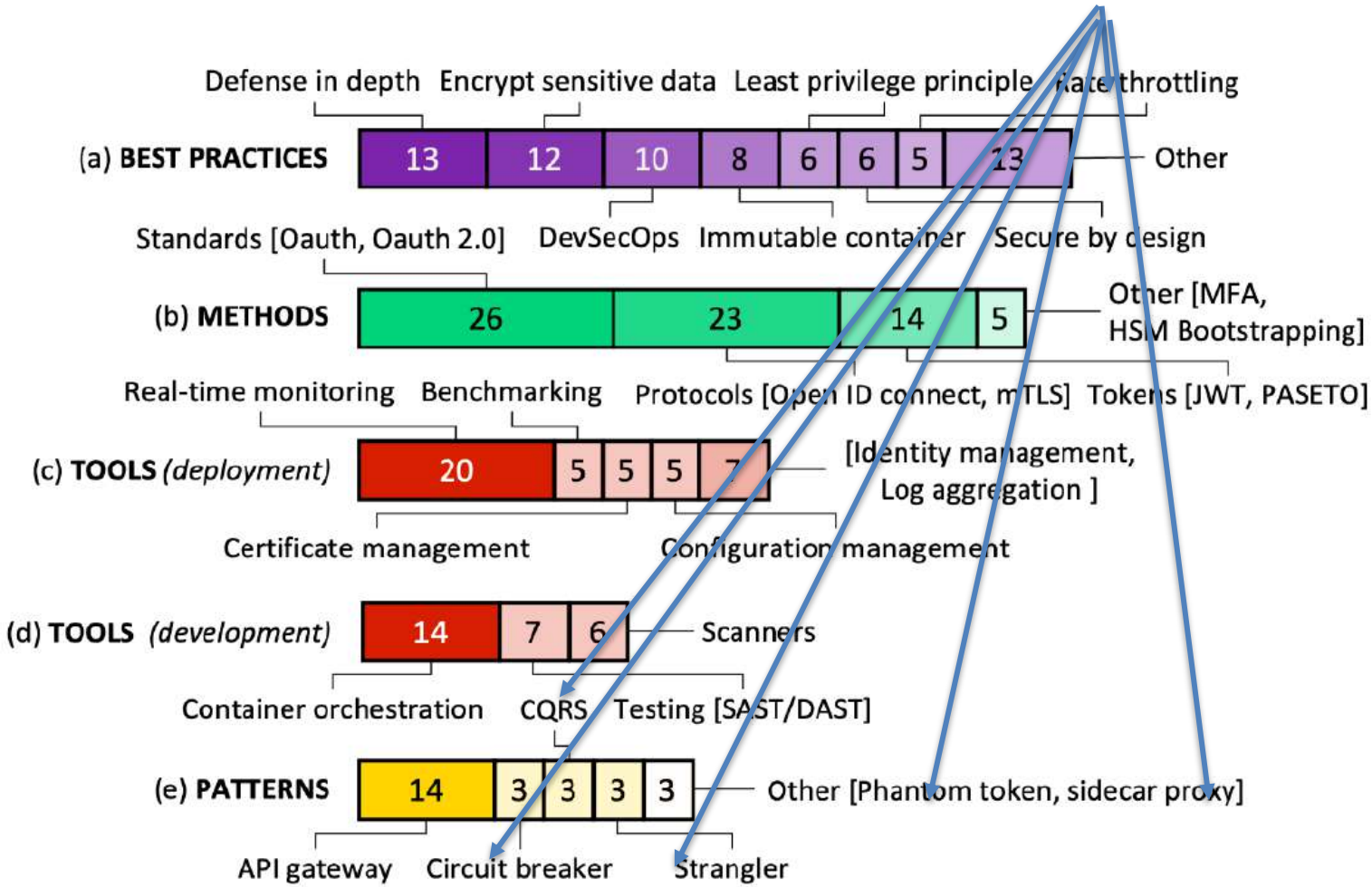
- Auditing the build process and at runtime
- Guaranteeing a clean container image is built on top of a trusted image
- Unnecessary components and libraries do not get bundled with the containers

HSM Bootstrapping

- HSM: Hardware security module
- Defend against attacks targeting hardware hosting the services and data.
- Also called trusted execution environments—which guarantee confidentiality and integrity of execution environments.

Similar to a secure enclave in mobile phone

Microservice security countermeasures



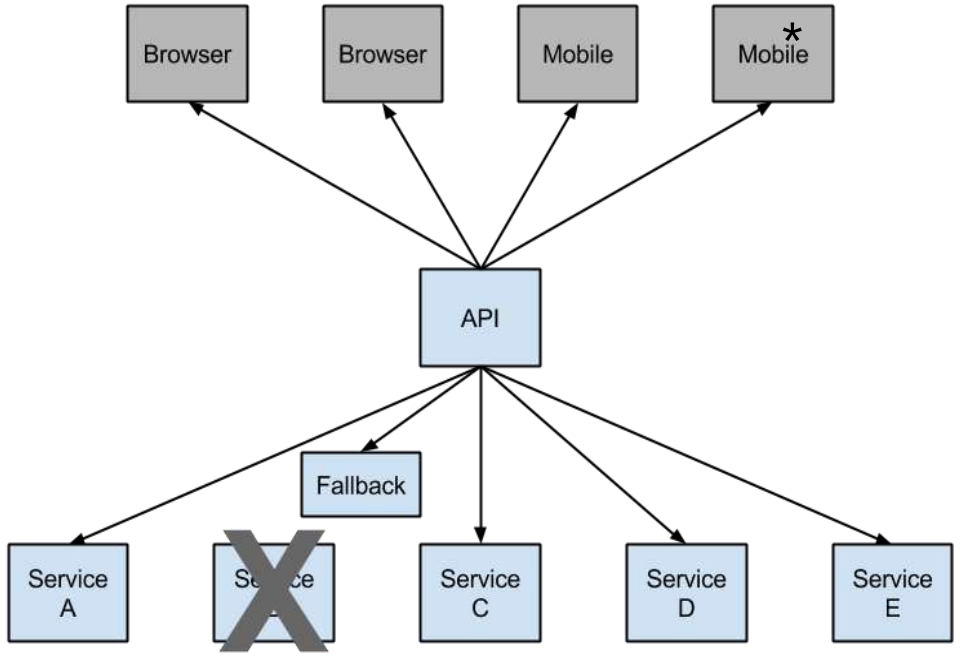
* Billawa et al. SoK: Security of Microservice Applications: A Practitioners' Perspective on Challenges and Best Practices (ARES '22).

Patterns

- Circuit breaker
- Command Query Responsibility Segregation (CQRS)
- Strangler
- Phantom token
- Sidecar proxy

Circuit breaker

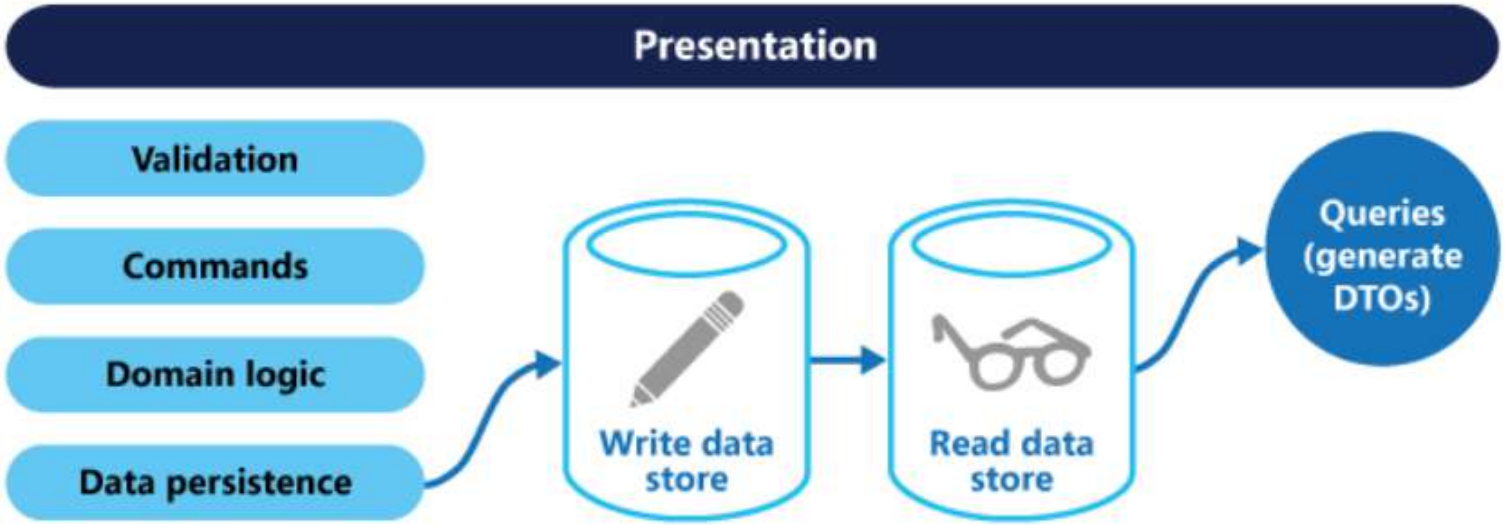
- Service failure protection and handle it so the failure will not propagate in the system.
- Real-time monitoring and alerting.
- Will tolerate the failures till a certain threshold after that, the fallback methods will be invoked.
- Gives a default behavior when services fail.



* https://dzone.com/articles/circuit-breaker-design-pattern-using-netflix-hystr

Command Query Responsibility Segregation (CQRS)

- CQRS separates read and update operations for a data store to optimize its performance, scalability, and security.
- **Security**. It's easier to ensure that only the right domain entities perform writes on the data.



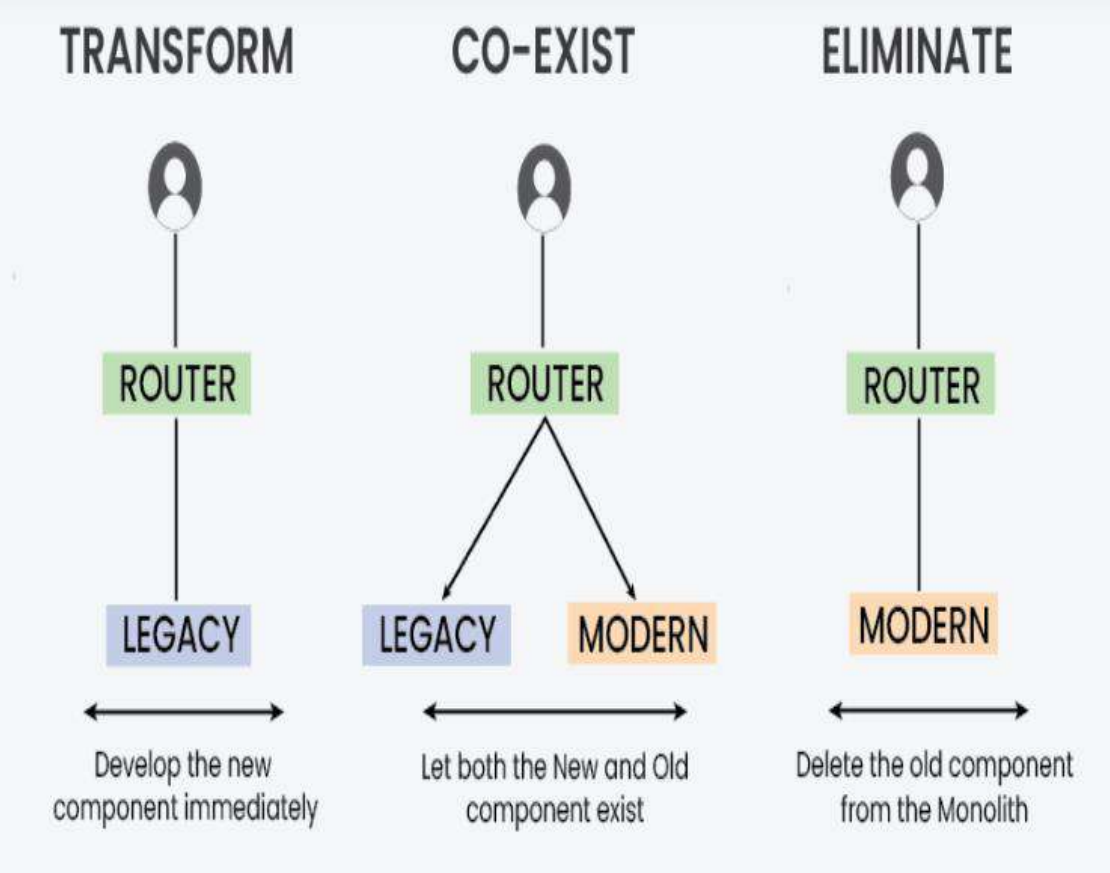
*

* https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs

Strangler

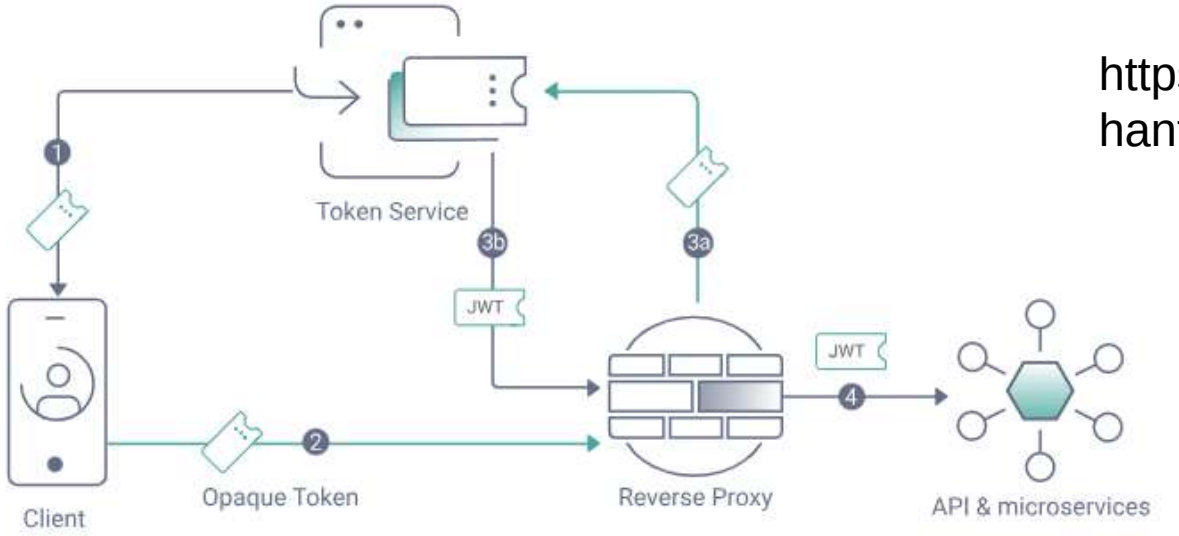
- Primarily used when migrating from a monolithic architecture to microservices.
- Mitigating risks associated with large-scale modernization projects.

*



* https://www.geeksforgeeks.org/strangler-pattern-in-micro-services-system-design/

Phantom token



https://curity.io/resources/learn/phantom-token-pattern/

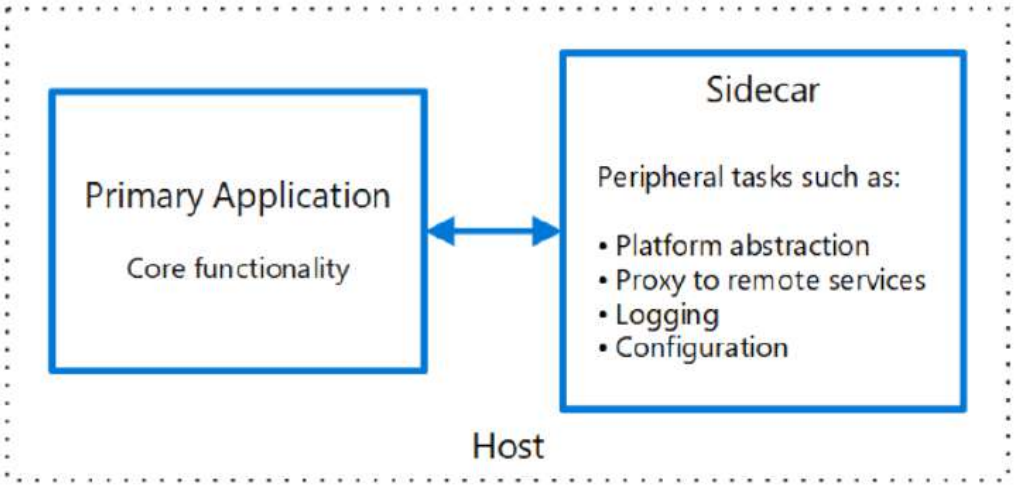
A combination of opaque and JWT tokens

1. The client retrieves an opaque token (random string).
2. The client forwards the token in its requests to the API.
3. The reverse proxy looks up the JWT token (containing information for authorization) by calling the Introspection endpoint of the Token Service.
4. The reverse proxy replaces the opaque token with the JWT token in the actual request to the microservice.

Sidecar proxy

- The sidecar is attached to a parent application and provides supporting features for the application.
- Co-locate a cohesive set of tasks with the primary application but place them inside their process or container.

You can also use sidecars to add cross-cutting security controls to an application component that is not natively designed with that functionality.



https://learn.microsoft.com/en-us/azure/architecture/patterns/sidecar

Summary

- Microservice architecture attack surfaces and countermeasures
  - Top-level service exposed to Internet
    - E.g., API-gateway
  - Load balancers
    - E.g., Rate throttling
  - Communication between microservices
    - E.g., service-level authorization, service-to-service authentication
  - Containers
    - E.g., secure container
  - Host hardware
    - E.g., HSM Bootstrapping