Microservices Security: Fundamentals and Best Practices

Anders Eknert

5 min read
November 17, 2022

The many benefits of microservices architecture, such as improved scalability and agility, explain why organizations are migrating from the traditional monolithic architecture. However, these benefits come at the cost of increased complexity, especially regarding security. According to an O'Reilly survey, 56% of respondents considered increased complexity the greatest challenge in developing microservices.

This article will help you understand the basics of securing microservices and the best practices for an effective security design.

Key takeaways:

— Microservices architecture consists of small loosely-coupled services that can function independently from each other.

— Microservice applications present a greater attack surface and may have hundreds of components that dev teams must adequately secure.

— Microservices architecture consists of small loosely-coupled services that can function independently from each other.

— Following all the security best practices, including policy-based controls for microservices APIs and users, allows you to maintain the integrity of your applications and reap the benefits of this cloud-native architecture.

What Are Microservices?

Microservices are small, independent services within an application, each performing a different function. These services communicate with each other using lightweight application programming interfaces (APIs). Their independent nature allows teams to work on the service they are responsible for without affecting the rest of the application. Data storage solutions can also differ for each microservice, along with the programming language used.

Unlike monolithic applications where a single process failure can cause the entire application to stop working, in a microservice architecture, if a fault is discovered in one service, the other application functions can continue working.

Microservice architecture is fast gaining popularity. Verified Market Research projects that the microservice market will reach $6.62 billion by 2030 — a 21.7% CAGR growth from 2023 to 2030.

How to Secure Microservices?

Even though we have seen the use of microservices spread rapidly, application security hasn't kept pace. There is no universal method of securing microservices. Instead, developers must consider several factors when designing security, such as risks and time-to-market, and find a balance between security and the organization's needs and resources.

Best Practices for Designing Microservice Security architecture

The attack surface of microservices is larger than that of monolithic applications, and more vulnerabilities can be associated with this architecture due to the number of components.

Here are seven best practices to protect the integrity of your application:

1. Use shift-left and DevSecOps strategies

Using a shift-left approach means that application security is kept in mind through all the stages of development. Testing and performance evaluations are done early in the lifecycle. Practicing DevSecOps promotes security skills and awareness within teams and ensures that security professionals are a part of the entire development process instead of only being consulted towards the end.

2. Make applications secure by design

To protect data, developers must integrate several layers of security into the application, prioritizing security in microservice application design, building and deployment phases. Prioritizing security means constantly testing the code and your continuous integration (CI) and continuous delivery/deployment (CD) pipelines.

Static analysis security testing (SAST) and dynamic analysis security testing (DAST) can be used for this purpose. Here's the difference:

—**SAST** uses a scanner compatible with your programming language to detect vulnerabilities in your code and the libraries used.

— **DAST** mimics an attack from the outside and does not need to be in a specific programming language.

3. Practice defense-in-depth

Just as castles have inner walls in case attackers breach the outer walls, the services containing the most sensitive data in your application should also have several layers of defense in place. Otherwise, attackers might be able to access them if they can exploit another service within the application.

Do not simply rely on a firewall to protect these services, for example. You should also use best access control practices, such as token-based identification and fine-grained policy-based authorization across users and services.

Other recommended security measures include data encryption and monitoring the application for suspicious activity.

4. User authentication and authorization

A key aspect of microservice security is protection against unauthorized access. It is necessary to use standards-based identity and access management (IAM) for user authentication, such as SAML, WS-Fed or the OpenID Connect/OAuth2 standards. Multi-factor authentication (MFA) should also be added as a security measure.

For authorization in microservice applications, you can use Styra Declarative Authorization Service (DAS) to implement and manage policy-as-code. As a control plane for OPA, Styra DAS allows you to enforce fine-grained control at scale without impacting latency and performance.

To learn more, check out our post on how to enforce fine-grained authorization in microservices.

5. Create API gateways

Many API gateways, such as the Amazon API Gateway, come with out-of-the-box management and security features and are often designed to scale. They take care of authentication and rate limiting for you and protect the microservice application from unauthorized access and large volumes of traffic used in distributed denial-of-service (DDoS) attacks.

As with most components of microservices, API gateways also need a way to implement authorization. Using Styra DAS to manage OPA, you can separate access logic from the application code to govern, monitor and audit all traffic flow.

6. Ensure container security

Developer teams often use containers for microservice applications to deal with many components and make deployments easier. Container security risks include compromised images, configuration and isolation faults and vulnerabilities in the host OS, which can compromise all the containers running on it.

Practicing the security principle of least privilege is recommended, along with the following strategies:

— Restrict authorization to the minimum requirements.

— Do not use Sudo or privileged accounts to run or launch services.

— Limit the access and consumption of available resources.

— Avoid storing secrets on container disks.

— Set rules to isolate access to resources.

Container orchestration tools, such as Kubernetes, allow developers to automate container scaling, deployment and security processes. Following best Kubernetes security and authorization practices can significantly speed up the deployment process. **For a fast and easy way to put guardrails around Kubernetes, you can deploy Styra DAS.**

Read our guide to Kubernetes compliance to learn about the regulatory requirements you should meet for increased container security.

7. Secure service-to-service communication

Microservices need secure means of communicating authentication and authorization requests with each other. Some of the methods used are:

— **Trust the network:** This outdated model has no security enforced into service-to-service communication and relies only on network-level security to prevent malicious attackers from communicating with microservices from outside.

— **JSON web tokens:** These are lightweight, self-contained tokens used to transmit the information as a cryptographically signed JSON object and provide authentication.

— **Mutual transport layer security (mTLS)**: Each microservice has a public/private key that provides two-way identification when communicating through mTLS.

— **Service mesh:** As the number of microservices within an application grows, developers may opt to implement a dedicated infrastructure layer, called a service mesh, that controls service-to-service communication. A service mesh adds a proxy sidecar to each microservice and typically uses mTLS to communicate with other proxies.

To solve the issue of authorization within a service mesh, you can use Styra DAS. With the native support of enterprise service meshes, Styra enables you to secure modern microservice applications with policy-enabled traffic control for real-time verification decisions. To see our solution in action, request a demo now.

Want to Learn More about Authorizing Microservices?

FAQs

Why is microservice security so important?

Microservices are susceptible to cyberattacks such as man-in-the-middle, injection attacks, cross-site scripting and DDoS, to name a few. They have a much larger attack surface than monolithic applications and it can be a challenge to aggregate security logs from many different services and platforms.

What are the benefits of a microservice deployment?

Breaking down applications into smaller independent components has numerous benefits, such as improved scalability, faster deployment and fault isolation. It also allows developers to use the programming language of their choice.

Several companies, including Amazon, Netflix, and eBay, have already embraced this software development strategy because of its many advantages.

What are some of the challenges of using a microservices architecture?

Microservices are more complex to build and deploy and have larger attack surfaces than monolithic applications. Inter-service communication can increase the latency demands and cause congestion in the network.

Since this is a new approach to software development, teams must learn to adapt and use a new set of tools, workflows and integrations.

Anders Eknert

Further reading

How to Enforce Fine-Grained Authorization in Microservices

Best Practices for Kubernetes Security

Get Ready For Next-Level Cloud-Native Authorization!

START FREE >

Speak with an Engineer

Request time with our team for a discussion that fits your needs.

SCHEDULE A DEMO >

Our mission is to provide unified authorization and policy across the cloud-native stack.

Sign Up for Our Newsletter

Company Email*

SIGN UP >

PRODUCT
Styra DAS
Styra Load
OPA Support
Open Policy Agent
Partners & Integrations
Styra DAS Free
Styra Load Free Trial

USE CASES
Application Authorization
Infrastructure Authorization
OPA Management

ABOUT US
Meet Styra
Leadership Team
Careers
Newsroom

DEVELOPER
Documentation
- Getting Started
- Tutorials
- Release Notes
- APIs

Styra Academy
- Terraform Validation
- OPA Performance
- OPA Policy Authoring
- Microservices Authorization

RESOURCES
Content Library
Blog
Academy
Events & Workshops