ZERO TRUST MADE PRACTICAL

4 Ways Policy as Code Enables Zero Trust in Microservices Applications

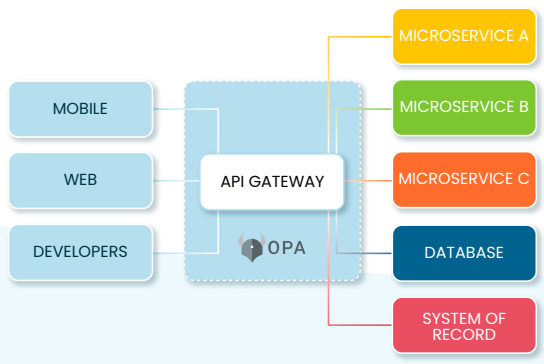
Enterprises are embracing cloud-native technologies and DevOps methodologies to deliver better software at greater efficiency and scale. Yet, the complexity of microservices applications and their environments exposes a significantly larger attack surface in applications — making a Zero Trust approach more critical than ever. Platform engineers must empower developers with tools and frameworks to make their applications secure. Here's how policy as code with Open Policy Agent (OPA) enables continuous authorization checks, based on contextual data, across a multitude of application components.

Zero Trust, per *NIST SP 800-207*, demands that every user or service must be both authenticated and authorized before accessing any resource.

1 Policy as code enables authorization checks for all North-South traffic at an external API gateway.

Policies:

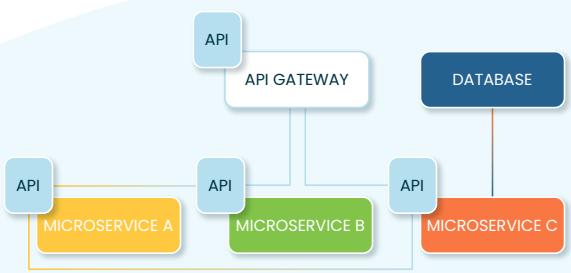
- Is User "Jane Fonda" allowed to access this service?
- Does every user have a valid cryptographic token?



2 It enables authorization checks for all East-West traffic at microservice APIs to ensure every action is authenticated and authorized at the last possible moment.

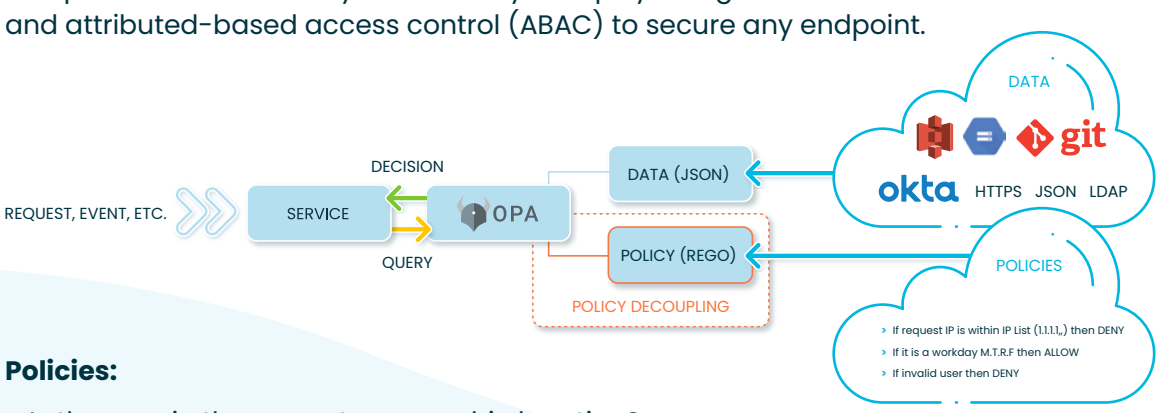
Policies:

- Is Service A allowed to talk to Service B?
- Is Service B allowed to fetch data from the backend database?
- Are all communications cryptographically signed with SPIFFE/SPIRE tokens?



3 Policy as code powers fine-grained AuthZ based on contextual and dynamic data, roles and attributes.

OPA provides the scalability and flexibility to deploy fine-grained role-based access control (RBAC) and attributed-based access control (ABAC) to secure any endpoint.



Policies:

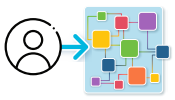
- Is the user in the correct geographic location?
- Is the Engineer currently on-call?
- Does the user control appropriate budget to be allowed access?
- Has the user been recently denied access for any other access policies?

4 Policy as code enables consistent policy lifecycle management, often using GitOps, and universal policy orchestration to enforcement points.

SERVICE-LEVEL POLICIES


```
# service attributes in header
allow {
   input.method == "GET"
   input.path == "/pets"
   input.header.source == "A"
   input.header.dest == "B"
}
```

USER-LEVEL POLICIES



```
# user attributes in authN token
allow {
   input.method == "GET"
   input.path == "/pets"
   input.token.claim == "customer"
}
```

EXTERNAL DATA DEPENDANCIES


```
# replicated ldap for employees
allow {
   some i
   data.ldap[input.user].role[i] == "admin"
}
# app calls OPA directly & overloads input
allow {
   some i
   data.ldap[input.user].role[i] == "manager"
}
```

- Promote policies through testing, staging and production pipeline stages — just like code.
- Create data and policy bundles and federate them to distributed enforcement points.
- Ensure end-to-end governance for every bundle for software supply chain security.
- Align on a universal standard to easily audit policy decisions for security and compliance.

Zero Trust is a journey, and it requires authentication and authorization. Only with policy as code, using OPA, can organizations apply true "never trust, always verify" policies at every critical access point throughout the layers of the cloud-native tech stack.

Want to learn more? Read our Deployment Architecture Guide: Implementing Microservices Authorization for Zero Trust Security