How We Use Istio and OPA for Authorization

Tao Lin, Software Engineer @HP
What is Authorization

Definition

Authorization is to verify if an identity has the permission to do something.
Common Approaches

**Role-based access control**
- Grant access by roles
- Coarse-grained
- Kubernetes RBAC, service account and role binding
- May cause role explosions

**Policy/Attributes-based access control**
- Grant access by policies
- Fine-grained
- AWS IAM Policies
- Policy is hard to write
FitStation Platform

A Custom Shoe Platform

Partners
- 3D Foot Scanning
- Shoe brands
- Shoe retailers
- Shoe manufacturers
- 3D shoe recommendation
- Sports league
High-level Architecture

Clients

L1

Partner

- Kiosk Service
- Vendor Service
- Manufacturer Service
- ...
- BackOffice Service

L2

Platform

- User Service
- Product Service
- Order Service
- Storage Service
- ...

L3

Internal

- Email Service
- Credentials Service
- Resource Service
- ...
- Templates Service
L1 Authorization

Requirements
- Enforced role-based access control in infrastructure layer

Design considerations
- L1 services has common requirements for role-based access control
- L1 services are mostly from 3rd party, we can’t enforce access control in code layer
RBAC enforced with Mixer
AuthZ Adapter

Adapter rule

```yaml
# rule to dispatch to the handler
apiVersion: "config.istio.io/v1alpha2"
kind: rule
metadata:
  name: authzrule
  namespace: istio-system
spec:
  match: destination.labels["layer"] == "L1"
  actions:
  - handler: handler.authz
  instances:
  - authzrequestcontext.authorization
```

- Services are labeled with the layer they’re in
- Only requests to the service with the layer equals “L1” will be handled by AuthZ adapter
AuthZ Adapter

Authorization template

```yaml
# instance configuration for authorization template
# compose the inputs for adapter using istio attributes
apiVersion: "config.istio.io/v1alpha3"
kind: authorization
metadata:
  name: authzrequestcontext
  namespace: istio-system
spec:
  subject:
    user: request.auth.principal | ""
    groups: request.auth.principal | ""
  properties:
    service: source.service | ""
    namespace: source.namespace | ""
  action:
    namespace: destination.namespace | ""
    service: destination.service | ""
    method: request.method | ""
    path: request.path | ""
    properties:
      token: request.headers["authorization"] | ""
```

- Leverage RESTful API in Permission definition. Each RESTful API as a permission
- Extracts token, service, namespace, method and path from istio.
L2 Authorization

Requirements
- Fine-grained access control
- Externalized authorization logic from business logic
- Dynamic and configurable access control

Design considerations
- L2 services are resources from different parties, resource access should be separated
- Different solutions may have different resource access policies
- Policies may change from time to time
ABAC enforced with OPA
Open Policy Agent

- A general-purpose Authorization engine that loads your policy data and rules and make a decision
- Policy data is JSON format files
- Policy rule is written in declarative language REGO
- Query data is a JSON payload
OPA Policy Example

GET hp-order-service.hp/orders?company_id=hp
Authorization: jwt_token

```
package fitstation.authz.order
import data.relations.employees

is_order_service {
  input.service_name = "hp-order-service.hp"
  allow {
    is_order_service
    input.method = "GET"
    input.resource_path = "/orders"
    input.user.roles[.] = "retailer-admin"
    employee = employees[input.user.user_id]
    input.query_string.company_id = employee.companies[.]
  }
}
```
Thank You!