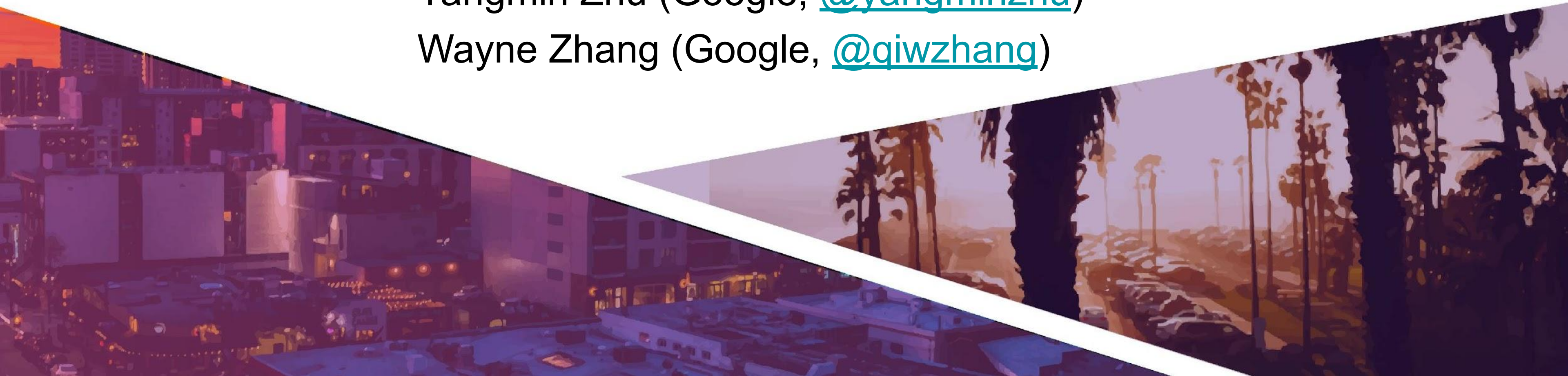
envoycon

Overview of Authentication and Authorization Features in Envoy

Yangmin Zhu (Google, @yangminzhu)
Wayne Zhang (Google, @qiwzhang)



Problem

Enforce access control based on identities or various other information in Envoy?

Solution

1) jwt_authn filter for authentication of JSON Web Token (JWT)

2) RBAC filter for authorization inside Envoy

3) ext_authz filter for authorization out of Envoy

Authentication - JWT

The jwt_authn HTTP filter verifies JSON Web Token(JWT)

It verifies JWT:

- Signature,
- Issuer,
- Audiences.

JWT payload output to:

- HTTP header for backend
- DynamicMetadata for other filters

Its filter config:

- Auth providers:
  - How to get jwks (public keys)
  - Where to extract token.
  - How to pass the JWT payload
- Requirement rules:
  - Which requests should be verified?
  - Which provider's token should be verified

Authentication - JWT

envoycon

jwt_authn filter config sample:

```
providers:
  provider_name1:
    issuer: https://example.com
    audiences:
    - bookstore_android.apps.googleusercontent.com
    remote_jwks:
      http_uri:
        uri: https://example.com/jwks.json
       cluster: example_jwks_cluster
  provider_name2:
    issuer: https://example2.com
    local_jwks:
      inline_string: PUBLIC-KEY
    from_headers:
    - name: jwt-assertion
    forward: true
     forward_payload_header: x-jwt-payload
```

```
Rules:
# /health doesn’t require verification
- match:
    prefix: /health
# /api paths use provider_name1 jwt
- match:
    prefix: /api
  requires:
    provider_and_audiences:
      provider_name: provider_name1
      audiences:
        Api_audience
# all other paths use provider_name2 jwt
- match:
    prefix: /
  requires:
     provider_name: provider_name2
```

Authorization - RBAC

Role Based Access Control (RBAC) Filter enforces access control inside Envoy.

- RBAC Filter = Action + A list of Policies
  - Action = Allow or Deny
  - Policy = Permissions + Principals + Conditions
    - Permission/Principal: built-in AST (and/or/not/etc.), fast but limited semantics
    - Condition: Common Expression Language, more flexible but slower
  - Shadow Policy: Evaluated and logged but not enforced, useful for testing
- Traffic: supports both HTTP or TCP
- Stats: allowed/denied/etc.

Authorization - RBAC

envoycon

Sample RBAC config

Action: ALLOW (whitelist)

One policy: `"product-viewer"`

Permission (ANDed):

- request with `GET` method
- path with prefix `/admin`
- port `80`

Principal (ORed):

- JWT token of subject `"admin"`
- x509 certificate of principal `"production"`

```
action: ALLOW
policies:
  "product-viewer":
    permissions:
    - and_rules:
        rules:
        - header: { name: ":method", exact_match: "GET" }
        - header: { name: ":path", prefix_match: "/admin" }
        - destination_port: 80
    principals:
    - or_ids:
        ids:
        - authenticated:
            principal_name:
              exact: "production"
        - metadata:
            filter: envoy.filters.http.jwt_authn
            path:
            - key: https://example.com
            - key: sub
            value:
              string_match:
                exact: admin
```

Authorization - ext_authz

External Authorization (ext_authz) Filter enforces access control out of Envoy.

- ext_authz filter = Service + Some other configurations
  - The Service specifies various information about the authorization service
    - Where to find it: cluster name
    - How to talk to it: gRPC or HTTP
    - What to be included in the request/response: could be used for token exchange
- ext_authz sends CheckRequest to the authorization service
  - Includes attributes of the source, destination and connection
- Supports both HTTP and TCP traffic
- Stats: ok/error/denied/etc.

Authorization - ext_authz

envoycon

Sample ext_authz config

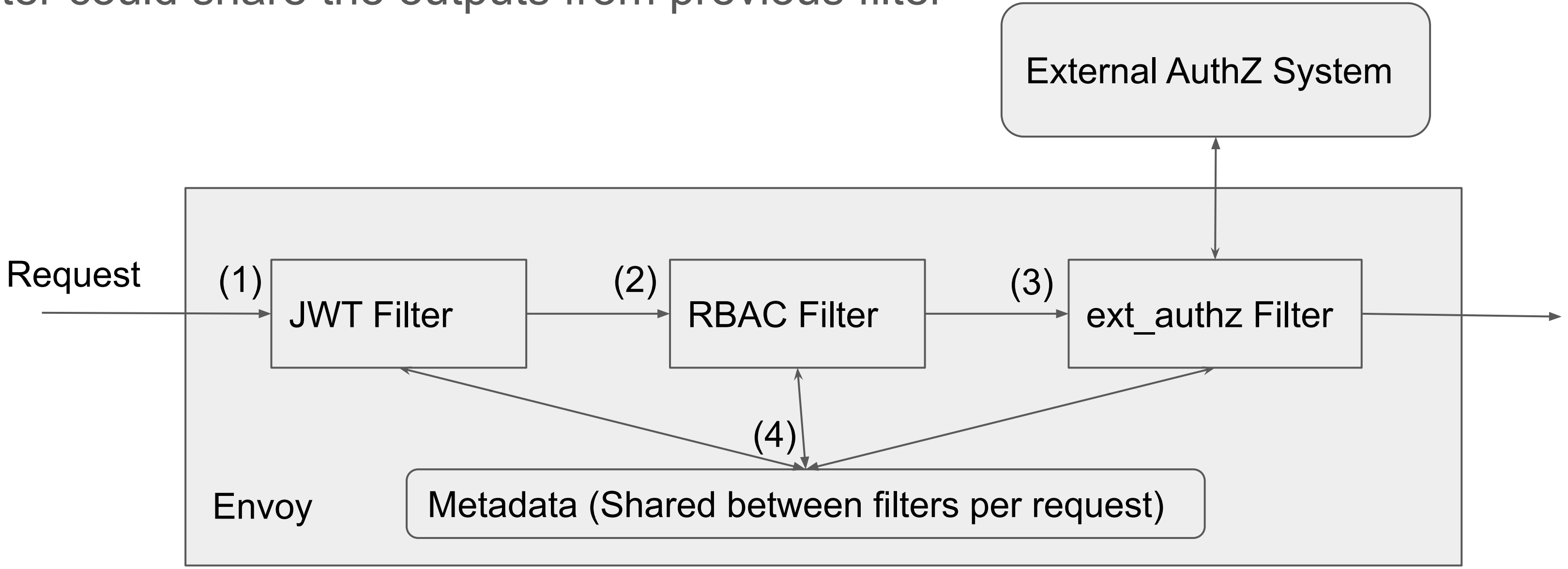
- Authorization service is defined as a HTTP service
- failure_mode_allow: false
  - Reject requests if the communication with the authorization service has failed
- Metadata_context_namespaces: `envoy.filters.http.jwt_authn`
  - Pass the jwt payload to the authorization server
- cluster ext-authz defines how to talk to the authorization service
  - Set **`tls_context`** to verify the authorization server and encrypt the traffic

```yaml
http_filters:
- name: envoy.ext_authz
  config:
    http_service:
      server_uri:
        uri: 127.0.0.1:10003
        cluster: ext-authz
        timeout: 0.25s
        failure_mode_allow: false
        metadata_context_namespaces:
        - envoy.filters.http.jwt_authn

clusters:
- name: ext-authz
  connect_timeout: 0.25s
  type: logical_dns
  lb_policy: round_robin
  load_assignment:
    cluster_name: ext-authz
    endpoints:
    - # Omitted
  tls_context:
  # Omitted
```

Example Use Case

(1) **jwt_authn** verifies and extracts JWT claims
(2) **RBAC** to enforce access control locally in Envoy
(3) **ext_authz** plugs in external authorization system
(4) Each filter could share the outputs from previous filter



Caveats

1. Make sure the filter is configured correctly for protecting your service
    a. Add tests for both positive (allow) and negative (deny) cases
    b. Be careful when upgrading Envoy if you’re using --ignore-unknown-field
2. Configure the JWT/RBAC/ext_authz in front of other filters
3. The jwt_authn filter supports both remote_jwks and local_jwks
    a. remote_jwks: Envoy needs to talk to external service to fetch the jwks once in a while
    b. local_jwks: You need to maintain the jwks manually for rotation
4. Use of RBAC or ext_authz
    a. RBAC is much faster but only supports built-in semantics
    b. ext_authz is much flexible but introduces an extra network request, use TLS to increase the security (verify the authorization server and encrypt traffic)
    c. Choose depending on your service QPS, goal and requirements