Lab 16 - Gloo Platform OPA Integration

OPA is an open source, general-purpose policy engine that you can use to enforce versatile policies in a uniform way across your organization. Compared to a role-based access control (RBAC) authorization system, OPA allows you to create more fine-grained policies. For more information, see the OPA docs.

OPA policies are written in Rego. Based on the older query languages Prolog and Datalog, Rego extends support to more modern document models such as JSON.

- Reminder to set the `GLOO_GATEWAY_HTTPS` environment variable

```
export GLOO_GATEWAY_HTTPS=$(kubectl --context cluster-1 -n istio-ingress get svc -l
istio=ingressgateway -o jsonpath='{.items[0].status.loadBalancer.ingress[0].*}'):443

echo "SECURE Online Boutique available at https://$GLOO_GATEWAY_HTTPS"
```

Native OPA Integration

Gloo Mesh's OPA integration populates an `input` document to use in your OPA policies. The structure of the `input` document depends on the context of the incoming request, described in the following table.

OPA input structure	Description
`input.check_request`	By default, all OPA policies contain an Envoy Auth Service `CheckRequest`. This object has all the information that Envoy gathers about the request being processed. You can view the structure of this object in the **attributes** section of the linked Envoy doc.
`input.http_request`	When processing an HTTP request, Envoy populates this field for convenience. For the structure of this object, see the Envoy `HttpRequest` docs and proto files.
`input.state.jwt`	If you use OAuth, the token retrieved during the OIDC flow is placed into this field.

- Create an OPA policy to be ready by Gloo Platform

```
cat <<EOF > policy.rego
  package test

  default allow = false
  allow {
      startswith(input.http_request.path, "/currencies")
      input.http_request.method == "GET"

  }
EOF
```

- Create configmap for the policy

```
kubectl create configmap allow-currency-admin --from-file=policy.rego --context
cluster-1 -n online-boutique
```

- Create an `ExtAuthPolicy` that validates incoming requests against the OPA policy

```
kubectl apply --context management -f - <<EOF
apiVersion: security.policy.gloo.solo.io/v2
kind: ExtAuthPolicy
metadata:
  name: api-auth
  namespace: app-team
spec:
  applyToDestinations:
  - selector:
      labels:
        app: currency
  config:
    server:
      name: ext-auth-server
      namespace: ops-team
      cluster: management
    glooAuth:
      configs:
      - opaAuth:
          modules:
          - name: allow-currency-admin
            namespace: online-boutique
          query: "data.test.allow == true"
EOF
```

- Test requests to the currency service

```
# get the available currencies NO API Key
curl -vk https://$GLOO_GATEWAY_HTTPS/currencies

# get the available currencies with API Key
curl -vk -H "x-api-key: developer" https://$GLOO_GATEWAY_HTTPS/currencies
curl -vk -H "x-api-key: admin" https://$GLOO_GATEWAY_HTTPS/currencies

# convert a currency with developer key
curl -k -H "x-api-key: developer" https://$GLOO_GATEWAY_HTTPS/currencies/convert \
--header 'Content-Type: application/json' \
--data '{
  "from": {
    "currency_code": "USD",
    "nanos": 0,
    "units": 8
  },
  "to_code": "EUR"
```

```
}'

# convert a currency with admin key
curl -k -H "x-api-key: admin" https://$GLOO_GATEWAY_HTTPS/currencies/convert \
--header 'Content-Type: application/json' \
--data '{
  "from": {
    "currency_code": "USD",
    "nanos": 0,
    "units": 8
  },
  "to_code": "EUR"
}'
```