Architecture & Prototype SMO Service
Exposure & Discovery in O-RAN Opensource

Using Keycloak & JWTs to expose & secure services used by rApps

OSC NONRTRIC

12 Oct 2022
Enforcing Service Exposure / Access

Why we need selective and controlled access to services

Securely enforce exposure without changing your code
What is Service Exposure? And why is it important?

— Apps (rApps) need to use SMO services, and provide service to other Apps

— Apps & SMO service may be multivendor

— We need *selective & controlled exposure* of these services

— New standardised *O-RAN R1 interface*
How to do Service Exposure?

- Ensure that services cannot be accessed without a carefully allocated ‘Token’
- Apps (consumers) are allocated a ‘Token’ when deployed or instantiated (and continuously updated)
- Without changing Provider or Invoker!
- Service Registration & Discovery (more on this later)
- Even more fine-grained access policies (more on this later)
What do we show here?

- API Gateway
- How to implement exposure (authorisation) Platform services (*provider*)
- How to apply exposure (authorisation) policies to rApp *µ*services (*provider*)
- How to continuously create Tokens for rApps (*consumer*)
- Use/Integrate 3GPP-spec’ed CAPIF core functionality for Registry/Discovery
- Use/Integrate OPA Policies for more fine-grained access
rApp request flow

- **1: Invoker rApp**
- **2: EnvoyFilter**
- **3: Jwt-proxy**
- **4: Istio Gateway**
- **5: Request authorization**
- **6: Provider rApp**

**Provider**

- Provider sends response back in the other direction (using reverse path – via Istio Gateway).

**EnvoyFilter**

- Request is intercepted by EnvoyFilter.

**Jwt-proxy**

- Jwt-proxy retrieves/reuses JWT from Keycloak.
- The request's header is updated to include the JWT.

**Request authorization**

- The JWT is checked using either the Istio internal authorization mechanism or an external authorizer like OPA.

**Gateway**

- Gateway routes the request to the appropriate service (with JWT included).

**Invoker**

- Invoker sends request to Provider (Addressed via Istio Gateway).

**Back to Provider**

- Provider sends response back in the other direction (using reverse path – via Istio Gateway).
Demo

```yaml
namespace: smo
releaseName: pmsnhs
repository:
  repoName: chartmuseum
protocol: http
address: 10.152.183.155
port: 88
userName: onapinititalizer
password: demo123456
overrideParams:
global.masterPassword: test

org.onap.domain.database.Local_HM: MicroserviceControlLoopElement:
  # Chart installation without existing repository
  version: 1.2.3
  type: org.onap.policy.clamp.controlloop.KBSMicroserviceControlLoopElement
  description: Control loop element for the KBS microservice control loop
  properties:
    provider: ONAP
    participantId:
      name: "participantName"
      version: 1.0.0
    participantType:
      name: org.onap.kbs.controlloop.KBSControlLoop
      version: 2.3.4
    chart:
      chartId:
        name: nginx-ingress
        version: 1.0.1
        releaseName: nginxapp
        namespace: smo
org.onap.controlloop.HttpControlLoopKBSparticipant:
  version: 2.3.4
  type: org.onap.policy.clamp.controlloop.PKBSparticipant
  description: Participant for HTTP requests
  properties:
    provider: ONAP
org.onap.domain.database.HTTP_PMSHMicroserviceControlLoopElement:
  # Consul HTTP config for MSG.
  version: 1.2.3
```
Bonus Topic

Standardising Service Exposure
3GPP CAPIF & O-RAN R1-SME

— We need to:
  — find services
  — register service providers (incl. rApps)
  — control service invokers (including rApps)

— 3GPP specifies CAPIF APIs for Service Registry

— O-RAN specifies R1 Service Catalog
Demo

```yaml
namespace: smo
releaseName: pjsons
repository:
  repoName: chartmuseum
protocol: http
type: org.onap.policy.clamp.controlloop.KBSMicroserviceControlLoopElement
version: 1.2.3
provider: ONAP
controllerId:
  name: controllerName
  version: 1.0.0
participant:
  name: org.onap.kbs.contro lle.KBSCanBeLooper
  version: 2.3.4
chart:
  chartId:
    name: nginx-ingress
    version: 0.9.1
  namespace: smo
org.onap.controlloop.HttpControlLoopElement:
  version: 2.3.4
provider: ONAP
org.onap.domain.database.HTTP_PMSHMicroserviceControlLoopElement:
  version: 1.2.3
```
Bonus Topic

Enforcing more fine-grained exposure / access
Even more fine grained — OPA policies

— “Open Policy Agent”

— OPA policy checks at service invocation-time

— OPA policies can be included with app/service deployment charts
— OPA policies can be hosted on a bundle server and injected into your app through a sidecar
Demo

```
namespace: smo
releaseName: pmsns
repository:
  repoName: chartmuseum
protocol: http
type: org.onap.pom.without.repos
port: 80
user: operatortester
password: demo123456
```

```
org.onap.domain.database.LocalAttachmentMicroserviceControlLoopElement:
  # Chart installation without a registry repository
  version: 1.2.3
type: org.onap.pom.without.repos
source: http://10.152.183.155
  type_version: 1.0.6
description: Control loop element for the K8S microservices.
  properties:
  - name: ONAP
    version: 1.0.0
  - name: org.onap.k8s.controlloop.K8SControlLoop
    version: 2.3.4
  chart:
    version: 0.9.1
    releaseName: nginxapp
    namespace: smo
    org.onap.controlloop.HttpControlLoopDefault:
      version: 2.3.4
type: org.onap.pom.controlloop.PolicyCondition
  description: Participant for http requests
  properties:
    - name: ONAP
      version: 1.2.3
```
Wrapping Up

Questions?