R1 PoC - R1 Interface exposure and JWT Authorization

This article contains a working PoC of R1 interface exposure of PMS and ICS with JWT Authorization.

The PoC is launched as VM in Vagrant/Virtualbox. The VM installs kubernetes and all other needed sw.

Scripts and kubernetes manifests are included.

Prerequisites

Virtualbox - Can be downloaded from this page Download Virtualbox

Vagrant - Can be from this page Download Vagrant

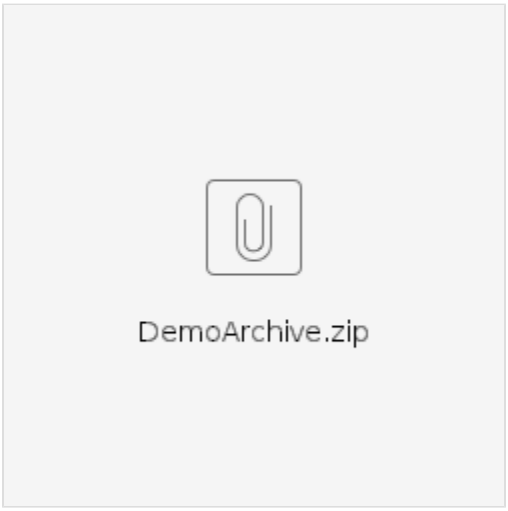
Tested on Mac with X86 HW.

It is also possible to run the PoC without Vagrant/VirtualBox on a Kubernetes cluster (docker-desktop, minikube etc) with Istio installed as well as with support for Network Policies (Cillium or Calico).

Preparations

Create a new dir under your "vagrant home" directory.

Download the Vagrant and demo zip files and put them in the new dir and unzip.



Setup the PoC

Open a shell (denoted shell1) on you host and in the dir where the Vagrant file is located, do:

shell1
$ vagrant up

Take note of the following line ip/port may vary):

Keycloak Admin Console: http://10.0.2.15:32131/auth/admin

Setup port forwarding, in the Virtualbox GUI, from a port on the local host (for example 2226) to the ip and port listed in the keycloak url.

Open the keycloak GUI in the browser (user/pwd is admin/admin)

Create a realm, client and user according to: https://www.keycloak.org/getting-started/getting-started-kube

Make sure to set "Temporary" to off when creating the users

- realm: nrtrealm
- client: nrtclient
- user: pmsuser (same pwd)
- user: icsuser (same pwd)

Take note of the "ID" (hexstring) for each user

Continue in shell1

shell1
$ vagrant ssh
$ sudo su
$ cd demo

Continue in shell1 and apply Network Policies to shield off the namespace nonrtric from all access from namespaces other than nonrtric and istio-system

shell1
$ kubectl apply -f np-nrt.yaml
networkpolicy.networking.k8s.io/np-nrt created

Start two pods. Will be used to inject rest calls from the default and the nonrtric namespaces to the simulated ICS and PMS services

shell1
$ kubectl apply -f upods.yaml
service/ubuntui created
pod/ubuntui created
service/ubuntui created
pod/ubuntui created

Start the test script - this script will try to make a number of rest calls to PMC and ICS and print the response.

Initially, all calls will fail.

Note that this script need to be restarted every 5min due to JWT token expiry.

shell1
$ clear; ./test.sh kc

Open a new shell (shell2) and login to the VM

shell2
$ vagrant ssh
$ sudo su
$ cd demo

Continue in shell2 and start PMS and ICS

shell2

$ kubectl apply -f svc-pod-nrt.yaml

service/pms created

deployment.apps/pms created

service/ics created

deployment.apps/ics created

Wait for the pods to reach running state

shell2

$ kubectl get po -n nonrtric

```
NAME                   READY   STATUS    RESTARTS   AGE
ics-5f477c5cf-mjzzd    2/2     Running   0          2m4s
pms-645ff9bdf7-zvbkn   2/2     Running   0          2m4s
ubuntui                2/2     Running   0          13m
```

Check the printout in shell 1. The first two calls shall not respond "OK 200" since calls to PMS and ICS from nonrtric namespace is granted. The third and fourth calls are blocked since the source comes from a blocked namespace. This is accomplished by the Network Policies applied earlier.

Continue in shell2 to expose PMS and ICS over the R1 (istio) gateway

shell2

$ kubectl apply -f vs_gw_nrt.yaml

gateway.networking.istio.io/pmsgw created

virtualservice.networking.istio.io/pmsvs created

gateway.networking.istio.io/icsgw created

virtualservice.networking.istio.io/icsvs created

Verify in shell1 that all calls via the R1 gateway (last 8 calls) respond "OK 200". The PMS and ICS are now exposed over R1 but without any access control.

Edit these two files:

2ra_pa_nrt_ics.yaml 2ra_pa_nrt_pms.yaml

Replace the hex id in the two files with the "ID" values from keycloak noted earlier for each user.

`requestPrincipals: ["http://keycloak.default:8080/auth/realms/nrtrealm/25703efe-ee32-46df-adc8-027231272e1a"]`

Continue in shell2 to apply authorization for PMS

shell2

$ kubectl apply -f 2ra_pa_nrt_pms.yaml

requestauthentication.security.istio.io/ra-jwt-pms created

authorizationpolicy.security.istio.io/ap-jwt-pms created

authorizationpolicy.security.istio.io/ap-jwt-pms2 created

Verify in shell1 that calls via the R1 gateway for PMS (last 8 calls) respond "OK 200" only for the case when the token is OK (fails for calls with no token and bad token

Continue in shell2 to apply authorization for ICS

shell2

```
$ kubectl apply -f 2ra_pa_nrt_ics.yaml

requestauthentication.security.istio.io/ra-jwt-ics created

authorizationpolicy.security.istio.io/ap-jwt-ics created

authorizationpolicy.security.istio.io/ap-jwt-ics2 created
```

Verify in shell1 that calls via the R1 gateway for ICS (last 8 calls) respond "OK 200" only for the case when the token is OK and the operation is GET (fails for calls with no token, bad token and with POST + ok token.

The test script in shell2 create files with the token and public keys in the files:

- .pms_token
- .pms_token_decoded
- .ics_token
- .ics_token_decoded
- .client-pub-key

Also the response for each OK call are put in files (id corresponds to the id of the call in shell1):

- .<id>call