Managed Root CA Bundle

Document Shared Externally

Original Issue: https://github.com/kubernetes/kubernetes/issues/63726
Author: Jiajie YANG
Date: 2020-05-11

Motivation

- Webhooks/API extensions:
  - Audit registration
  - Admission registration
  - API registration

  Replace CABundle -> CABundleReference would make the config highly portable
  https://github.com/openshift/enhancements/pull/250
- Container trust bundles
- mTLS Service Meshes need bundle distribution
- Scalability
  - Root-ca-cert-publisher use ConfigMap
    Avoid copying the cert into every namespace's configmap

Requirements

- Implement a basic mechanism for cluster administrators to manage trusted CA bundle in cluster level. They could maintain the CA bundle used within the cluster by updating specific config in a single place.

Considerations

How do we inject the CA bundle?

There is no clear preference between these two options, so welcome comments on pros & cons for each option.

Option 1: Projected Volume
We provide the ability for cluster administrators to inject the CA Bundle object as a projected volume cluster-wide, and application operators could mount the volume to a specific path for libraries to retrieve the root CA bundle.

Option 2: ClusterConfigMap
Create a new cluster-wide ConfigMap, which is a new key-value pair object, to store the CA bundle data. All pods in the cluster would be able to read (no RBAC check shall be involved) this cluster-wide ConfigMap to retrieve the trusted CA bundle.

What should be the scope of the injected CA bundle?

We tend to make the injected CA bundle cluster-scope instead of namespace-scope.
(Tim Allclair has made an attempt at https://github.com/kubernetes/kubernetes/pull/67620 to push master certs to ConfigMap of a certain namespace)

- A cluster-scope trust distribution mechanism would make it easier to dispatch a default set of CA bundles for some of the use cases.
- From a scalability perspective, cluster-scope would save some space. Each CA bundle takes 1~400KB, copying CA bundles to multiple namespaces would take up significantly more storage for configs.

How do we safely roll out a change to a referenced CA Bundle?

Since we want to make the CA bundle as a reference instead of copying the actual certificate, there should be a safe roll-out plan when updating the bundle. There might be unexpected pod config referencing the CA bundle getting interrupted when the CA cert itself gets revoked/updated by cluster administrator.
Maybe it would be an overshoot to develop a general deployment mechanism. Considering documentation for some manual deployment process? (process like: inject a new CA bundle_v1.1, update mount reference so that pods use new cert after restart, then remove previous bundle_v1.0)

Where do we mount the default CA bundle cross OS/platform?

Mounting/referencing some default cluster-wide CA bundle into containers would relieve the burden of common users writing config about the CA bundle. However, the location of the default file retrieving root CA bundle might be different for each OS distribution (for defaulted certs we want to inject).
Since we might have containers with mixed OS distribution within one cluster, we need more thoughts on how we make sure all supported OS distributions would receive the projected CA bundle at their default location.

- Shall we mount the volume to different locations based on OS? How do we achieve that?
- We should not overwrite the existing CA bundle in the container, instead put our injected bundle alongside with the one carried in the container distro.

Example location of system-default CA bundle:
(from go library trying to locate system CA bundle)

Linux (try read file)

```
"/etc/ssl/certs/ca-certificates.crt"      # Debian/Ubuntu/Gentoo etc.
"/etc/pki/tls/certs/ca-bundle.crt"        # Fedora/RHEL 6
"/etc/ssl/ca-bundle.pem"                  # OpenSUSE
"/etc/pki/tls/cacert.pem"                 # OpenELEC
```

"/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem" # CentOS/RHEL 7

unix (try read all files under dir)
"/etc/ssl/certs",   // SLES10/SLES11, https://golang.org/issue/12139
"/system/etc/security/cacerts", // Android
"/usr/local/share/certs",   // FreeBSD
"/etc/pki/tls/certs",   // Fedora/RHEL
"/etc/openssl/certs",   // NetBSD
"/var/ssl/certs",   // AIX

Windows loadSystemRoots() seems not working as expected currently. Issue:
https://github.com/golang/go/issues/16736