Kubernetes Container Identity Working Group Turndown

Shared Publicly

Authors: gcastle@google.com, mikedanese@google.com
Updated: 2018-10-23

Goals and Scope

This doc explains why we're turning down the K8s Container Identity Working Group and includes a brief snapshot of some of the identity progress we made over the last 14 months as measured against the original goals.

It isn't an identity roadmap, or an exhaustive list of identity features we need to build.

Background

The Kubernetes Container Identity Working Group was founded on Aug 7, 2017 with Clayton defining the original Goals and Mission as follows:

Goals: It must be possible:

1. For containers to establish their *identity* via various means to other actors
2. For Kubernetes administrators to ensure that rogue containers can have their identity revoked
3. For existing security tools and solutions in wide use to integrate, to link authorization to external systems, or to provide assistive credentials
4. For container identity to be layered on top of Kubernetes instead of deeply coupled to it
5. For higher level concepts, like service identity, to also leverage or align with this approach

Mission: The Container Identity Working Group (WG) will bring together the appropriate experts inside the Kubernetes community (sig-auth, sig-node, sig-networking primarily, with sig-cluster-lifecycle as well) and outside the Kubernetes community (groups like SPIFFE or Istio) to move the discussion forward on:

1. How to allow containers to prove their identity

2. How to integrate existing process identity systems (certificate, kerberos, bearer tokens, proxies) with the goals above
3. The extensions in Kube required for this to be possible
4. How the installation and configuration of a Kubernetes cluster can result in this identity being trusted

Another set of detailed goals were proposed in this doc. In the interests of keeping this short we won't evaluate against those as well, but it would be a good exercise when planning future identity work.

What did we achieve?

The following is a brief snapshot of some of the identity progress we made over the last 14 months as measured against the original goals.

For containers to establish their *identity* via various means to other actors. The addition of TokenRequest and extension of TokenReview APIs means that the Kubernetes service account can be used to implement authentication beyond the Kubernetes API. A workload (daemonset, sidecar, or the application itself) can get a token from the K8s API server with an audience of an external consumer (e.g. myvaultinstance.company.com) that can use TokenReview to validate the token. The remaining work to complete this token-based flow is:

- Complete Audience support in TokenReview
- Graduate TokenRequest* features to GA
- More documentation and/or a codelab to make it easy for integrators to use.

It's possible to write a certificate approver to issue workload identity certificates via the K8s certificates API, but it's not a well documented or supported path. People considering this path have other options such as Istio or other service mesh frameworks which come with significant non-security advantages.

For Kubernetes administrators to ensure that rogue containers can have their identity revoked. For Kubernetes service accounts using the existing workflow the only revocation mechanism available is to delete and re-create the service account. For Kubernetes service accounts using the TokenRequest/Review workflow the lifetime of the tokens can be set as short as the integrator is willing to live with for availability. Revocation and refresh is still available via delete/re-create of objects bound to the token.

For existing security tools and solutions in wide use to integrate to link that authorization to external systems or provide assistive credentials. External systems integrators can choose to either:

1. Recognize K8s service account identity tokens natively and address them directly in authorization rules in external systems. e.g. Integrator recognizes K8s service accounts and you can write authz rules in terms of myk8snamespace/myk8ssa identities; or
2. Map them to “native” identities and write authz rules in terms of those “native” identities. e.g. Integrator can authenticate K8s service accounts and map myk8snamespace/myk8ssa to a myidsystem.myk8snamespace.myk8ssa identity, and authz rules are written in terms of that myidsystem.myk8snamespace.myk8ssa.

For this to be layered on top of Kubernetes instead of deeply coupled to it. K8s service account identity was kept separate from the credential delivery mechanism. Work is underway to migrate from storing forever-keys in K8s secrets, to shorter-lifetime JWTs created by the kubelet via the TokenRequest API.

For higher level concepts, like service identity, to also leverage or align with this approach. Istio is using the Kubernetes service account as the identity inside its x509 certs, identified with a SPIFFE prefix.

Why turndown this working group?

We're far from finished with identity work, but at this stage it doesn't make sense to have the overhead of a separate meeting and working group when the content can be handled in sig-auth.

Reflecting on the need for the meeting and the group itself:

- The OIDC work and tokenvolume have addressed many of the immediate pain points, and are largely in execution phase. The working group was useful for high bandwidth discussion early when the solution space was much more ambiguous.
- Current engagement in the meeting is low and it's essentially the same audience as sig-auth. Recently Google has been largely setting the agenda with little outside input/interest. There’s enough room in the sig-auth agenda to fit in the identity topics.
- There turned out to be less cross-cutting engagement than originally envisioned. There were a small number of discussions with storage and networking, but not enough to justify the extra meeting slot.