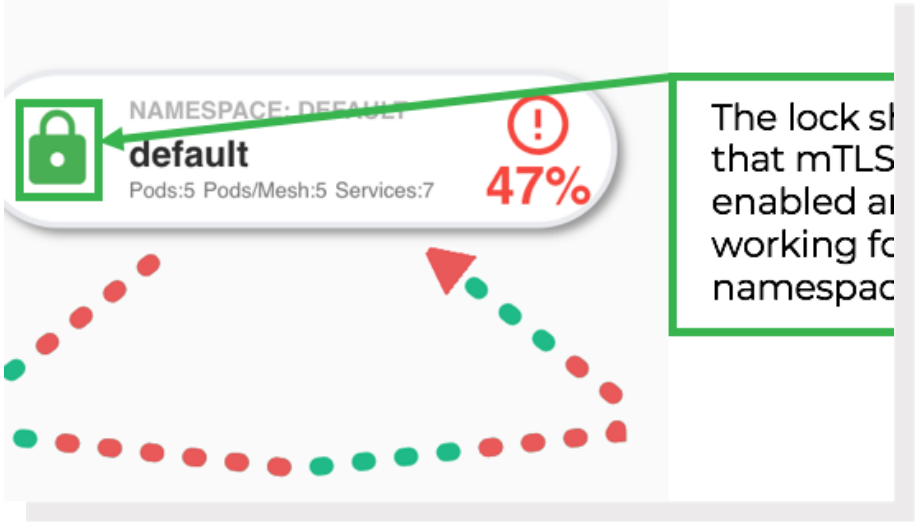
January 4, 2019

It’s 2AM: Do you know what your Istio Auth



Policies are doing?

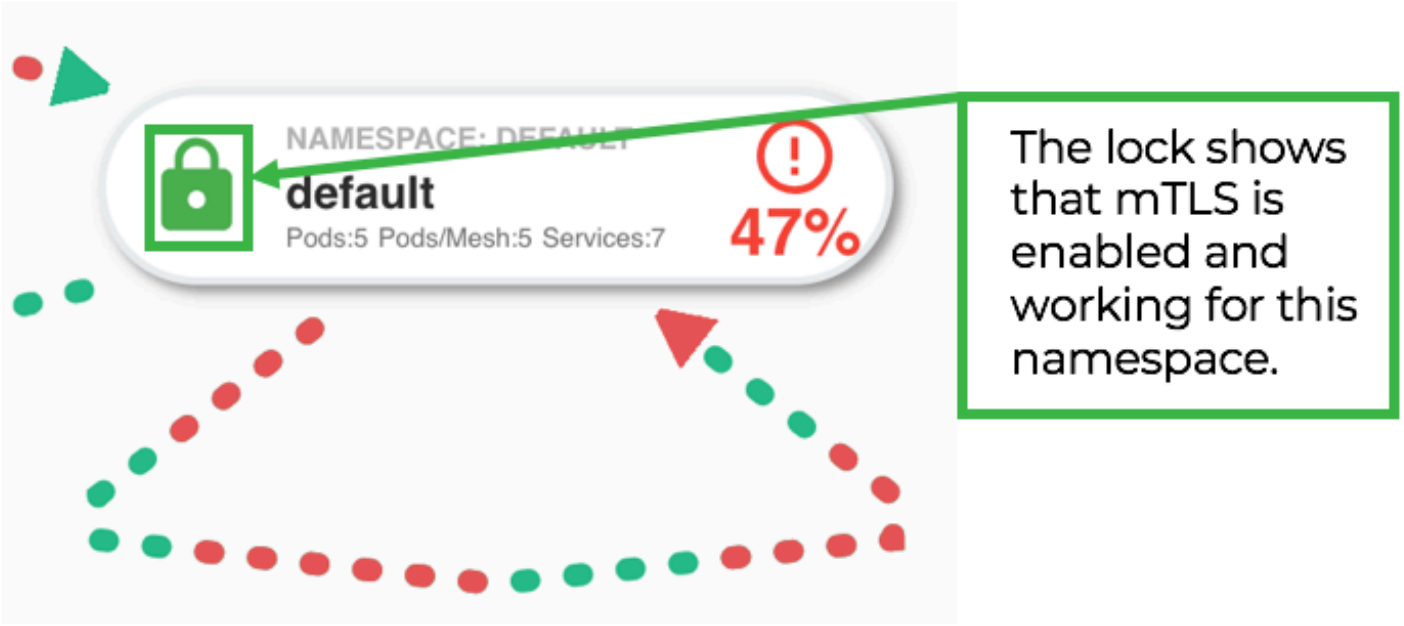
Lauren Billington

This year's KubeCon conference in Seattle was electrifying. There is a ton of energy around Kubernetes, and with the increasing interest and adoption, service meshes are gaining recognition as the way to better manage Kubernetes clusters at runtime. Istio was one of the most popular topics – in break-out sessions and on the expo floor.

It was exciting to get pulled aside in the hallways and crowded at our booth to answer questions about working with Istio, and ways that Aspen Mesh is addressing some of the hurdles of using a powerful but complex open source tool. One of the solutions I was excited to discuss was our Istio vetters, and how we make sure our users know their Authorization Policies and configurations are correct.

In addition to our open source vetters, we have built a UI that surfaces mTLS status at a glance, showing settings for different resources in your clusters. Since security is a critical issue, it's important to ensure certainty of encryption status — and that's not

always easy when you're using a quickly evolving solution like Istio.



With the changes between Istio's 0.8 release and 1.0, Istio added Mesh Policies as a new CRD. In addition, different ways of enabling a set of resources to receive mixed traffic was introduced (both enabled and disabled) were introduced. This mixed setting seems to be a work-around for users trying to migrate to mTLS-enabled traffic and is likely to be retired in future versions.

With the changes brought about by Mesh Policies and the partially-enabled mTLS settings, and the dearth of documentation on all the possible combinations, we needed to make sure that we knew all the combinations of Mesh Policy, Auth Policies, their targets, names, and what they actually did.

The process to understand all the possibilities involved setting up a cluster with various implementations of Istio demo apps, then creating the various resources (Mesh Policy, Auth Policies, Destination Rules, and some namespaces and services) and finally proceeding through a grid of combinations and curling to specific endpoints to see if the traffic was accepted or rejected. We also checked that Auth Policy settings were inherited by other resources as expected (ex: a service inherits the policy of the Mesh Policy when no policy is defined for it or for its namespace), and checked out what happens when two conflicting policies are written for the same target (indeterminate behavior!).

You can do it yourself, I've documented the process here, or you can save yourself some time and just reference the results as needed.

You will not need to go through this set up process if you are using Aspen Mesh as we have tested the different combinations and baked the correct configurations into Aspen Mesh. And, rest assured, we have your back when the next evolution of Istio features presents a new set of challenges!

Results

The following policies were implemented, then tested using the set up referenced above. They have been divided into 3 sections to help you determine what your policy has actually implemented.

Note that Auth Policies are shown, but the using the same config for Mesh Policies produced the same results.

MTLS ENABLED

The following examples of valid Auth Policies result in services that allow only authenticated traffic.

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "default"
  namespace: "foo"
spec:
  peers:
  - mtls: {}
```

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "default"
  namespace: foo
spec:
  peers:
  - mtls:
      mode: STRICT
```

MTLS DISABLED

The following examples of valid Auth Policies result in services that allow unauthenticated traffic.

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "default"
  namespace: "foo"
spec:
```

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "default"
  namespace: "foo"
spec:
  peers:
```

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "default"
  namespace: "foo"
spec:
  peers:
  - mtls:
```

MTLS MIXED

The following examples of valid Auth Policies result in services that allow authenticated and unauthenticated traffic.

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "default"
  namespace: "foo"
spec:
  peers:
  - mtls: {}
  peerIsOptional: true
```

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "default"
  namespace: "foo"
spec:
  peers:
  - mtls:
      mode: STRICT

  peerIsOptional: true
```

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "default"
  namespace: foo
spec:
  peers:
  - mtls:
      mode: PERMISSIVE
```

If you're using Istio and having any trouble managing your mTLS settings, hopefully this blog proves helpful. And if you simply don't want to deal with this, or other Istio complexity, Aspen Mesh simplifies Istio installation and management. Get it for free here.

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

POST COMMENT

< Back to Blog

Let us take the burden out of managing microservices

GET ASPEN MESH

hello@aspenmesh.io