
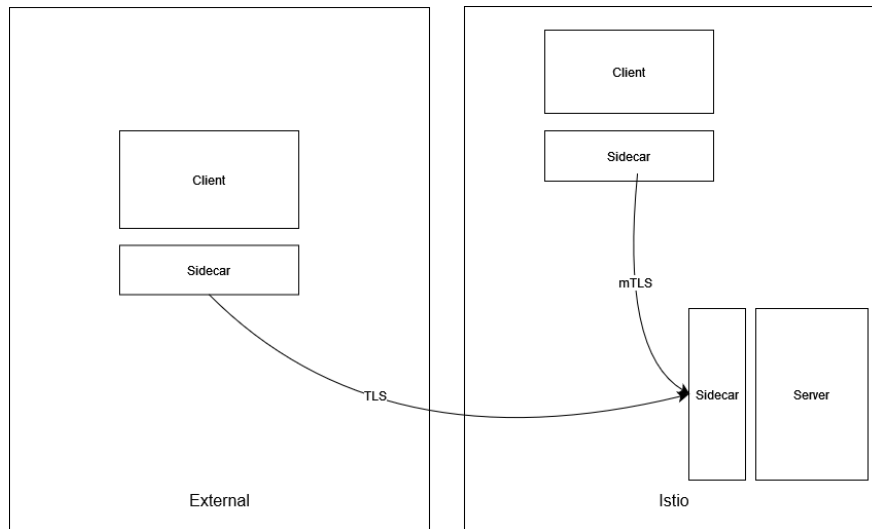


Istio RFC: Serve mTLS/TLS on the same port

Shared with Istio Community

	
Owner: William Zhang Working Group: networking	Status: WIP In Review Approved Obsolete Created: 2024-06-30 Approvers:

Objective



Currently, sidecars can be configured to terminate TLS with externally-provisioned certificates. However, doing so necessitates explicitly disabling internally-managed mTLS¹ on that port. This feature request proposes allowing for the termination of mTLS using Istio ALPNs and mTLS/TLS using other ALPNs on the same port.

Background

External services need a way to secure their traffic with services deployed within the Istio mesh. This is typically done through an additional proxy hop to an Istio gateway that serves as a bridge between the two. However, the additional hop introduces multi-tenancy that may not be tolerable from a latency and security standpoint. [Istio RFC: Support hybrid sidecar mode](#)

¹ E.g. provisioned by Istiod as a certificate authority, Istio-CSR

Shared with Istio Community

addresses the problems associated with the additional hop by allowing configuring sidecars to terminate TLS with certificates not minted by the sidecar itself.

Unfortunately, the existing implementation requires the PeerAuthentication mode to be explicitly set to “DISABLE”. As this prevents the sidecar from terminating user TLS and mTLS on the same port; to serve traffic originating from services both within and outside the mesh, two ports must be defined. This increases friction for cluster operators as they must make both ports discoverable and duplicate policies across both ports. This increases friction for service owners as they must select which port to use based on where their service is deployed.

Example Use Cases:

- Direct communication between Istio and external workloads
 - Hybrid cloud
 - Edge-to-core
 - SPIFFE Federation
- Gradual migration to Istio by seamlessly allowing servers to onboard prior to all their clients onboarding²

Requirements

- **Preserve security semantics:** For example, traffic should be rejected from external sources when configured to do so (e.g. through a PeerAuthentication config), irrespective of whether mTLS/TLS is used

Design Ideas

- Generate another inbound listener filter chain on the same port with a TlsInspector match, but exclude the ALPN match
 - This allows for the Istio mTLS filter chains to match on internal traffic as it is more specific
- Generate the filter chain only if the PeerAuthentication mode is DISABLE or PERMISSIVE. E.g.
 - **Disable**

```
filter_chains:
- filter_chain_match:
  transport_protocol: "tls"
  destination_port: 443
...
```

² Author's main use case

- **Permissive**

```
filter_chains:
- filter_chain_match:
    transport_protocol: "tls"
    application_protocols: ["istio-http/1.0", "istio-http/1.1",
"istio-h2"]
...
- filter_chain_match:
    transport_protocol: "tls"
    destination_port: 443
...
```

- **Strict**

```
filter_chains:
- filter_chain_match:
    transport_protocol: "tls"
...
```

Alternatives Considered

Alternatives ordered by descending preference

Maintain Two Ports

1. Define a new inbound listener reserved for user-TLS traffic that binds to another port but routes the traffic to the port listened by the service via defaultEndpoint
2. Expose the new port to service discovery (e.g. Kubernetes Service, ServiceEntry)
3. Define client-side VirtualService rules to route traffic to a “generic” port to ports reserved for Istio traffic

EnvoyFilter Patch on user-TLS Listener

We can perform the same operations listed in [istio/istio#53469](#) by defining a Sidecar configuration with a faux inbound listener and patching the port with an EnvoyFilter.

The below are the pros and cons of the listed options:

Unified Port (RFC)	Two Ports	EnvoyFilter Patch
Pros <ul style="list-style-type: none"> • Simplifies the mesh operator's work with less configurations • No change to applications Cons <ul style="list-style-type: none"> • Risky change to Istio source 	Pros <ul style="list-style-type: none"> • Most predictable behavior • Works out-of-the-box • Decoupling between the two virtual listeners Cons	Pros <ul style="list-style-type: none"> • Temporarily achieves the desired outcome of this RFC • User assumes all risk that this RFC may introduce Cons

<ul style="list-style-type: none"> code • Must consider interactions with existing and potential features (e.g. RBAC, HBONE) • Strong coupling³ between the two virtual listeners 	<ul style="list-style-type: none"> • Operationally complex; requires multiple configuration changes across different levels of the stack 	<ul style="list-style-type: none"> • Not maintainable • Not forward compatible • Breaks abstraction and error-prone; for this reason, strongly not recommended • Strong coupling between the two virtual listeners
---	---	--

³ For example, the ports field in an [AuthorizationPolicy](#)