


# Istio RFC: Support hybrid sidecar mode

Shared with Istio Community

	
<b>Owner:</b> <a href="mailto:shriram_sharma@intuit.com">shriram_sharma@intuit.com</a> <a href="mailto:anil_attuluri@intuit.com">anil_attuluri@intuit.com</a> <b>Working Group:</b> networking	<b>Status:</b> WIP   In Review   <b>Approved</b>   Obsolete <b>Created:</b> 2021-09-01 <b>Approvers:</b>

## Objective

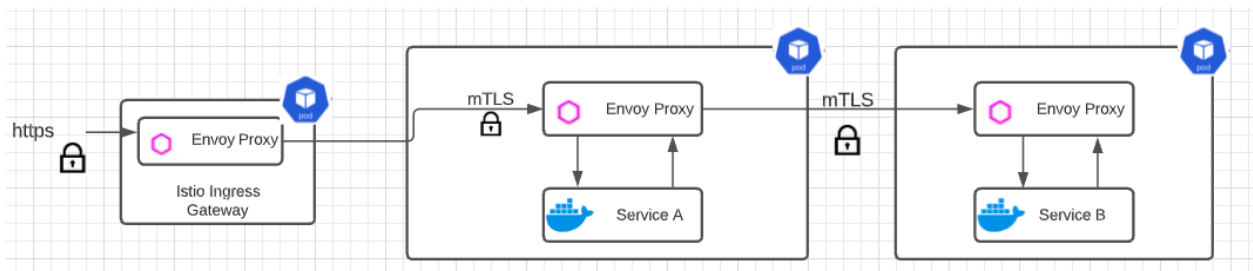
Enable TLS termination on the envoy sidecar. Currently, the server side envoy proxy running as a sidecar can only allow mTLS for secure connection. In a regular Istio mesh deployment the onus of one way TLS termination for downstream requests is on the ingress gateway. Though this satisfies most of the use cases, for some use cases (like an API Gateway on mesh) the ingress gateway resource is not necessarily needed.

The feature request is to introduce the “tls” property in the sidecar API, similar to gateway, which would allow the Istio proxy to run in a hybrid mode, i.e., it performs TLS and mTLS termination for requests originating from outside of the mesh.

The objective of this document is to propose a strategy for incorporating support for this behavior within Istio.

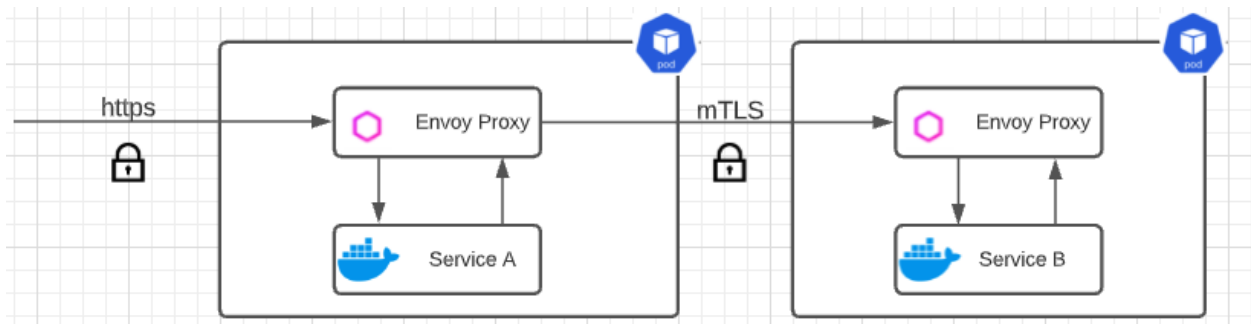
## Background

In most of the Istio Service Mesh deployments the expectation is to perform TLS termination on the Ingress Gateway pod. This pod in itself runs an Envoy proxy in “router” mode that can be configured to listen on an https port to perform TLS termination and then establish an mTLS connection to a backend service and route the request to it.



Shared with Istio Community

This design works for most use cases but in some cases there could be a need to eliminate an additional hop introduced by the Istio Ingress Gateway and let the Envoy sidecar running alongside the application perform TLS termination for requests ingressing from outside the mesh.



Currently, the istio-proxy container runs the Envoy Proxy either in router mode, for ingress gateway or in sidecar mode when running as a sidecar alongside the application. In **sidecar** mode the init container sets up all the required iptable rules and the control plane builds all the necessary listeners, routes, cluster and endpoint configurations and pushes them to the proxies.

Whereas in **router** mode, there are no iptable rules configured and the control plane builds listener, router, cluster and endpoint configurations based on the Gateway and VirtualServices resources.

In order to provide a way where a cluster can benefit from avoiding to configure an ingress gateway and the envoy proxy's functionality is extended to perform both as a router (for TLS termination) and as sidecar (to establish mTLS connection and route within the mesh), we extended the sidecar API by adding a new property, "tls", to the IstioIngressListener.

This feature is guarded by the following feature flag that can be set to 'true' to enable it.  
SIDECAR\_IN\_HYBRID\_MODE: true

## Requirements

- Istio should provide a way to perform TLS termination directly on the sidecar without having to add an ingress gateway.
- Istio should provide a way for the user to configure which deployment would enable its sidecar proxy to perform TLS termination.

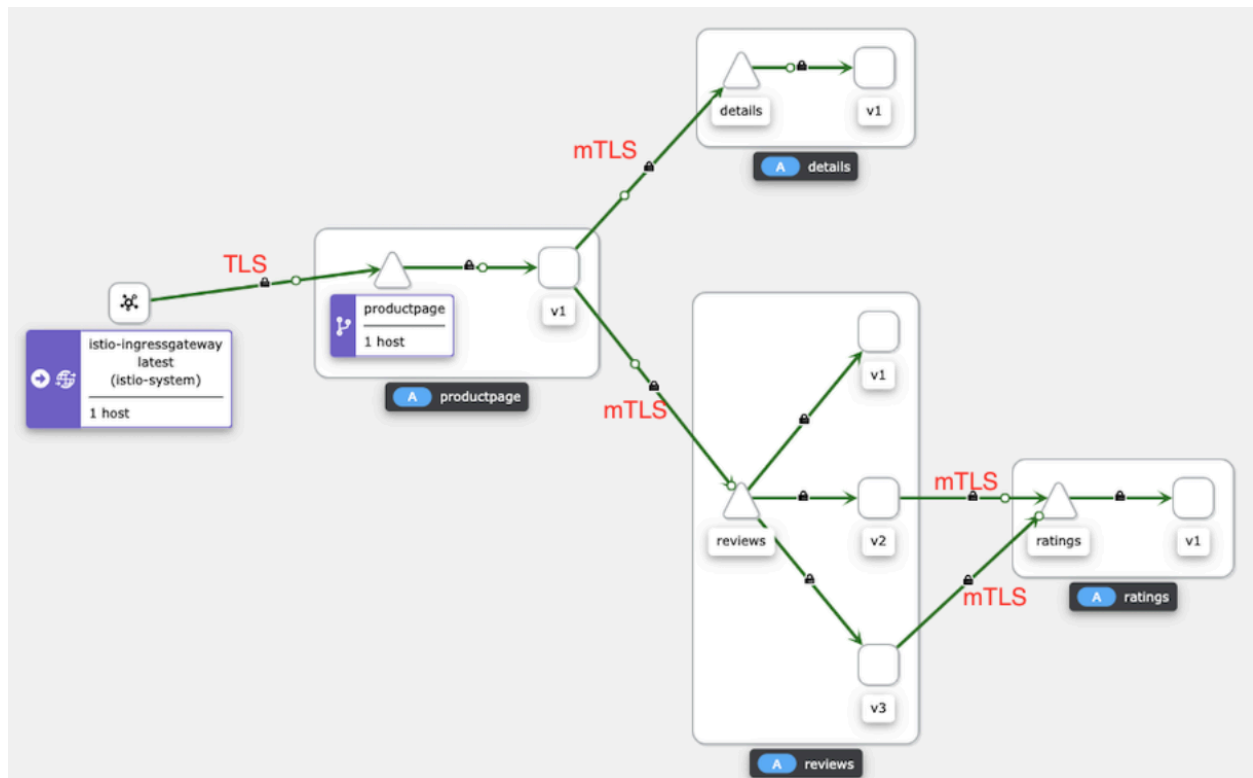
## Design Ideas

Add a property “tls”, to the sidecar API’s IstioIngressListener which if configured, will allow TLS/mTLS termination on the sidecar proxy with custom certificates.

### Example:

Let's take an example using the sample bookinfo application

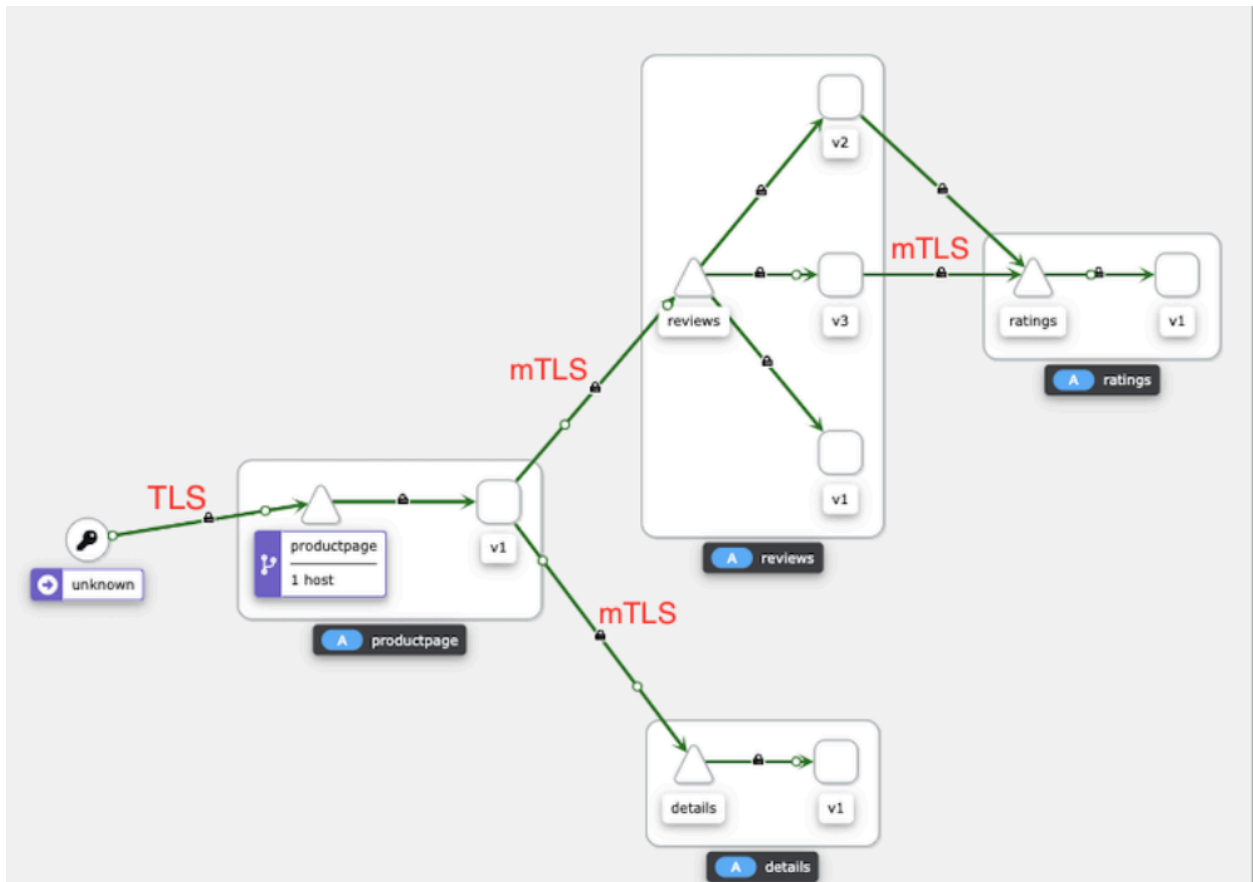
The graph below depicts the current Istio Service Mesh implementation where the TLS termination happens on the istio ingress gateway.



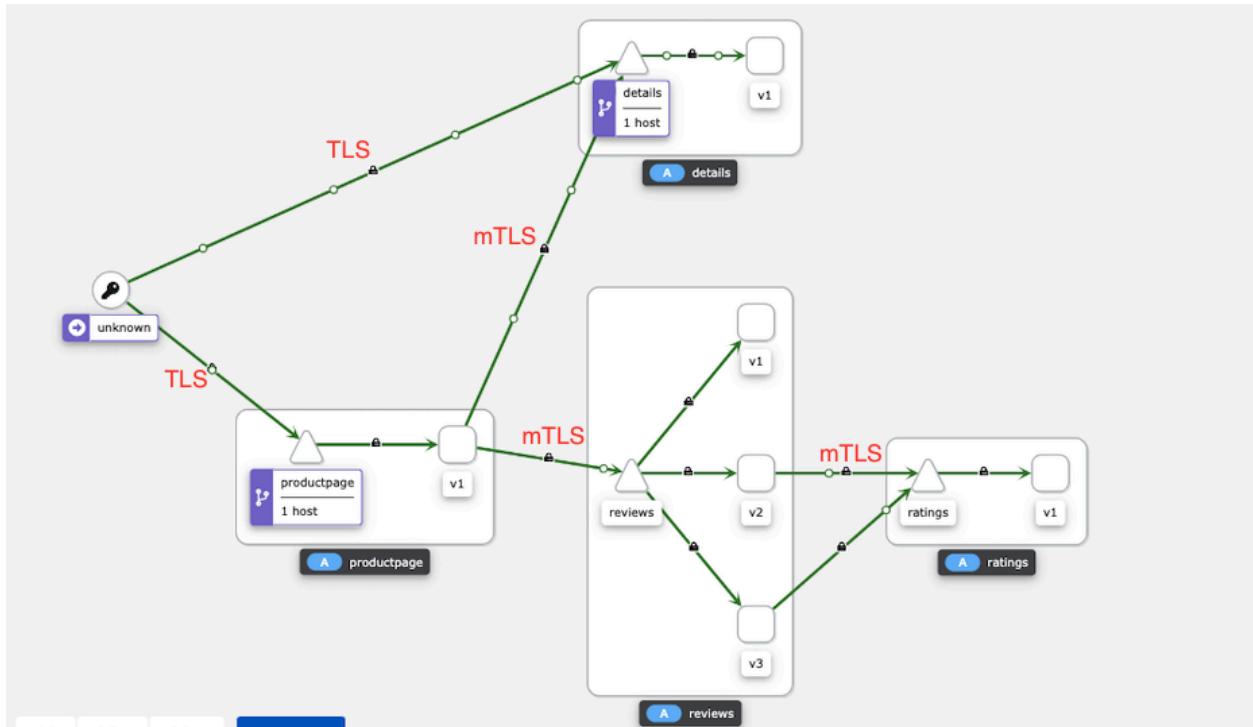
In the above example, the envoy proxy running inside the istio-ingressgateway resource is running as a “router” and performs TLS authentication.

The proxies running alongside the applications are running in “sidecar” mode and they establish mTLS connection with the upstream services before routing the requests.

By running the envoy proxy in productpage deployment in hybrid mode, we can eliminate the istio-ingressgateway hop. The https request originating from outside the mesh will be able to securely connect to the proxy directly.



The details service's istio-proxy is running in hybrid mode which TLS termination from calls outside of the mesh and mTLS when being called from the "productpage" app



In order to configure the proxy to run in hybrid mode, the following is needed.

1. Run the istio-proxy with environment variable "SIDECAR\_IN\_HYBRID\_MODE=true"
2. Configure the TLS settings as shown in the example below

```
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  labels:
    app: httpbin
    service: httpbin
spec:
  ports:
    - name: https
      port: 8443
      targetPort: 80
  selector:
    app: httpbin
---
```

```
apiVersion: networking.istio.io/v1alpha3
kind: Sidecar
metadata:
  name: httpbin
  namespace: app1
spec:
```

```

workloadSelector:
  labels:
    app: httpbin
ingress:
  - port:
      number: 80
      name: http
      protocol: HTTPS
    defaultEndpoint: 127.0.0.1:80
    tls:
      mode: SIMPLE
      privateKey: "/etc/certs/tls.key"
      serverCertificate: "/etc/certs/tls.crt"
---

apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: httpbin-pa
  namespace: app1
spec:
  selector:
    matchLabels:
      app: httpbin
  mtls:
    mode: STRICT

```

## °Challenges/Concerns

One of the main concerns that this implementation poses is that if due to misconfiguration, multiple deployments in a mesh run in hybrid mode, then we might unintentionally provide direct access to requests originating from outside the mesh.