# Advanced Application Management Using Red Hat OpenShift Service Mesh

Security

## Module Topics

- Access Control
- Istio Security
- Istio Identity
- Istio PKI
- Istio mTLS
- Authentication Policy
- Origin Authentication
- Authorization

## Access Control

- Can *entity* perform *action* on *object*?
- Authentication:
    - Verify credentials of entity
    - Ensure entity identity is valid
- Authorization:
    - Determine actions entity can perform

## Access Control

Authentication

- Transport Authentication
  - Service-to-service authentication
  - Verifies direct client making connection
  - Example: mTLS
- Origin Authentication
  - End-user authentication
  - Verifies original client (end user, device) making request
  - OpenID Connect with JSON Web Tokens (JWT)

## Istio Security

Goals

- Security by default
    - No changes needed for application code and infrastructure
- Defense in depth
    - Integrate with existing security systems to provide multiple layers of defense
- Zero-trust Network
    - Build security solutions on untrusted networks

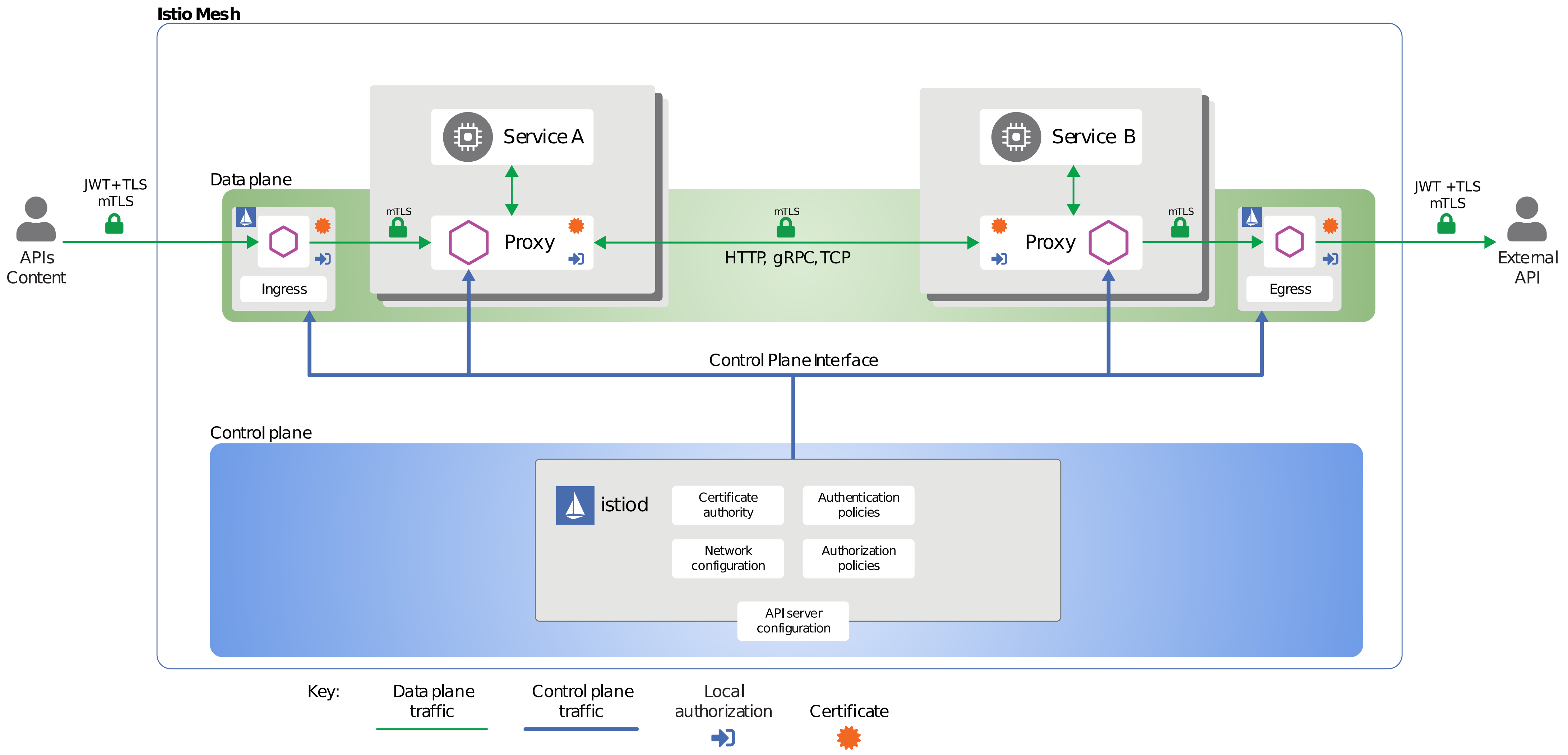
## Istio Security

Components

- Strong identity
- Security policies
- Transparent TLS encryption
- Authentication
- Authorization
- Auditing

# Istio Security

## Architect Diagram



## Istio Identity

- Identity is fundamental aspect of any security infrastructure
- Secure service-to-service authentication
  - Services exchange credentials with identity information
  - Client side:
    - Checks server identity
    - Verifies if it is authorized runner of service
  - Server side:
    - Checks client identity
    - Determines what client is authorized to do based on authorization policies

## Istio Identity
Model

- Based on SPIFFE standard
  - "Secure Production Identity Framework for Everyone"
  - Set of open source standards for securely identifying software systems in dynamic and heterogeneous environments
- Istio identity:
  - Based on Kubernetes service account
  - spiffe://<domain>/ns/<namespace>/sa/<serviceaccount>

## Istio PKI

- Built on top of Istiod Citadel component
- Securely provisions strong identities to every workload
- Uses X.509 certificates to carry identities in SPIFFE format
- Automates key and certificate rotation

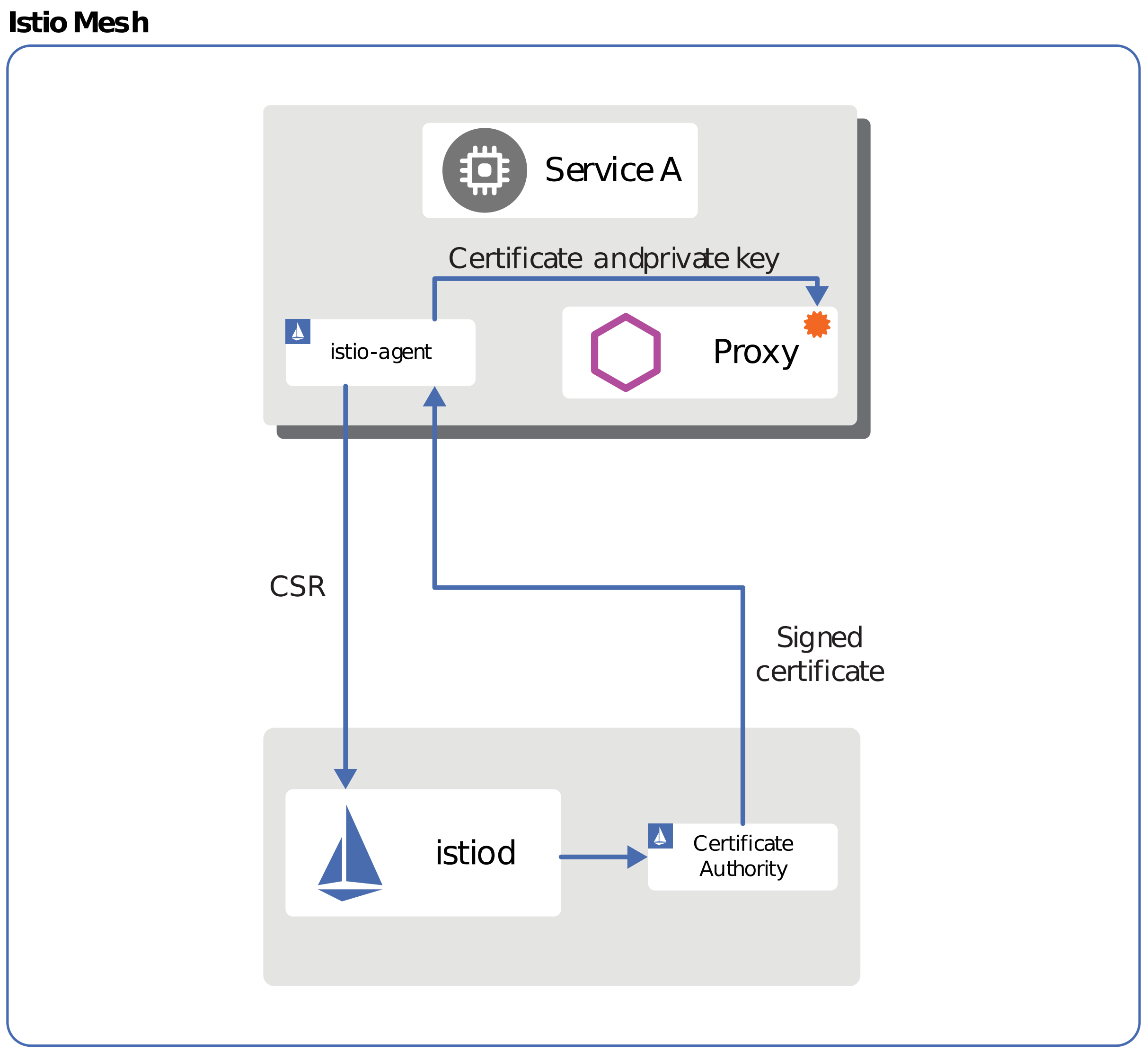
## Istio PKI

### Istio Agent and SDS

- SDS: Secret Discovery Service
  - Remote protocol for Envoy proxy to fetch certificates
- Istio agent: gRPC service that takes CSR requests
- When pod is created, Envoy sends certificate and key request via Envoy SDS API to Istio agent
- Istio agent creates private key and CSR, sends CSR with its credentials to Istiod for signing
- Istiod validates credentials carried in CSR, signs CSR to generate certificate
- Istio agent sends certificate received from Istiod and private key to Envoy via Envoy SDS API
- Process repeats periodically for certificate and key rotation

# Istio PKI

Istio Agent and SDS



## Istio mTLS

- mTLS provides two-way authentication and encryption
- With Istio, services can engage in mTLS communication
    - No need to implement mTLS at application level
- For client to call server with mutual TLS authentication:
    - Istio reroutes outbound traffic from client to client’s local sidecar Envoy
    - Client-side Envoy starts mutual TLS handshake with server-side Envoy
    - During handshake, client-side Envoy also does secure naming check to verify that service account presented in server certificate is authorized to run target service
    - Client-side Envoy and server-side Envoy establish mutual TLS connection, and Istio forwards traffic from client-side Envoy to server-side Envoy
    - After authorization, server-side Envoy forwards traffic to server service through local TCP connections

## Authentication Policy

- **CRD:** PeerAuthentication
- **Example:**

```
apiVersion: "security.istio.io/v1beta1"
kind: "PeerAuthentication"
metadata:
  name: "example-peer-policy"
  namespace: "foo"
spec:
  selector:
    matchLabels:
      app: reviews
  mtls:
    mode: STRICT
```

- **Policy scope: Service-specific > namespace-wide > mesh-wide**
- **mTLS values:** STRICT, PERMISSIVE

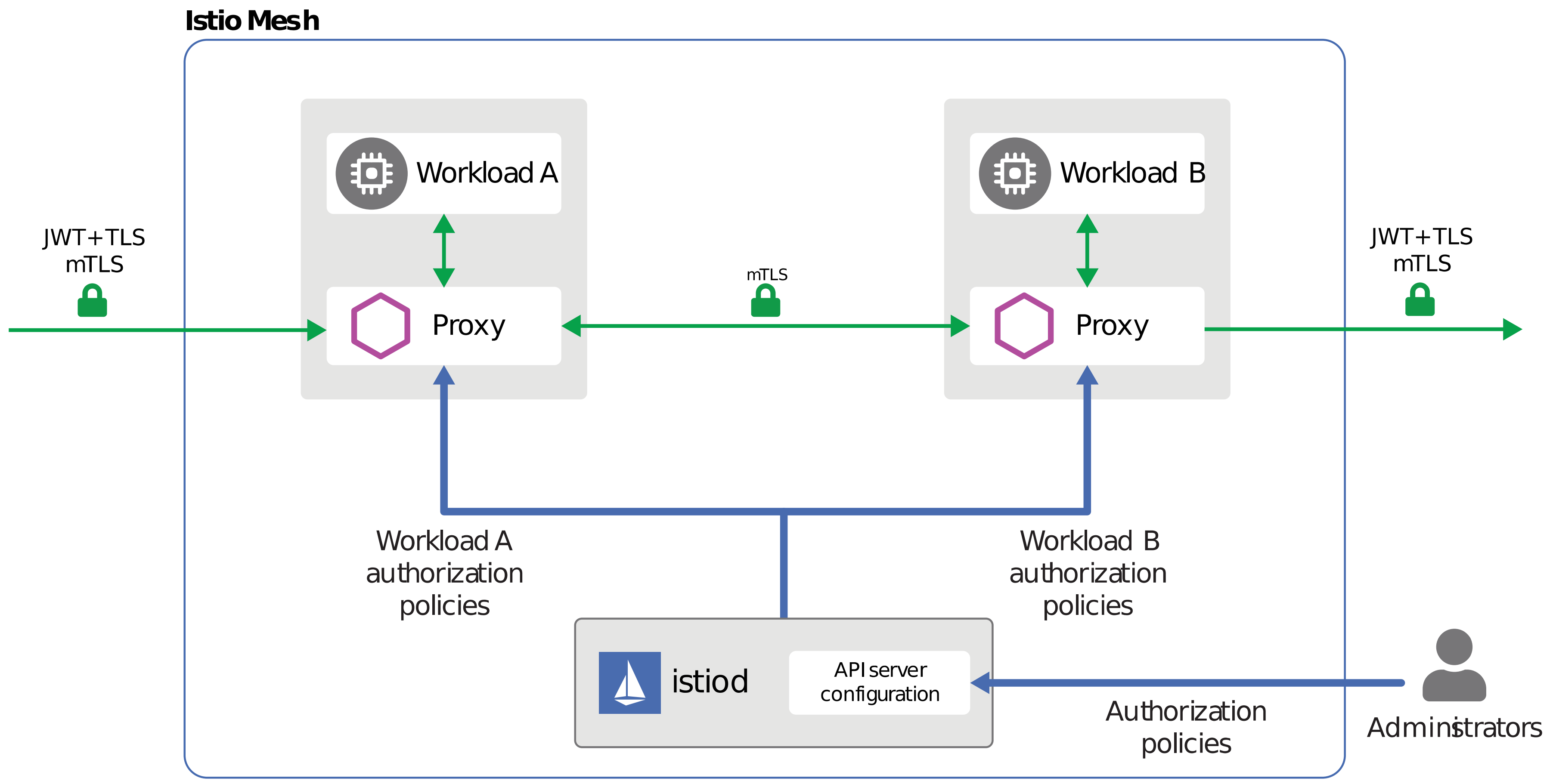
## Origin Authentication

- CRD: RequestAuthentication
- Istio supports JWT authentication only
- JWT token passed to application by service proxy
- Application responsible for JWT token propagation

```
apiVersion: security.istio.io/v1beta1
kind: RequestAuthentication
metadata:
  name: ingressgateway-origin
spec:
  selector:
    matchLabels:
      app: istio-ingressgateway
  jwtRules:
    - issuer: https://accounts.google.com
      jwksUri: https://www.googleapis.com/oauth2/v3/certs
```

# Authorization

- CRD: AuthorizationPolicy
- Workload-to-workload and end-user-to-workload authorization.



## Authorization

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
 name: httpbin
 namespace: foo
spec:
 selector:
   matchLabels:
     app: httpbin
     version: v1
 action: ALLOW
 rules:
 - from:
   - source:
       principals: ["cluster.local/ns/default/sa/sleep"]
   - source:
       namespaces: ["dev"]
   to:
   - operation:
       methods: ["GET"]
   when:
   - key: request.auth.claims[iss]
     values: ["https://accounts.google.com"]
```

## Authorization

Authorization on plain TCP protocol

```
apiVersion: "security.istio.io/v1beta1"
kind: AuthorizationPolicy
metadata:
  name: mongodb-policy
  namespace: default
spec:
 selector:
   matchLabels:
     app: mongodb
 action: ALLOW
 rules:
 - from:
   - source:
       principals: ["cluster.local/ns/default/sa/bookinfo-ratings-v2"]
   to:
   - operation:
       ports: ["27017"]
```

## Module Summary

- Access Control
- Istio Security
- Istio Identity
- Istio PKI
- Istio mTLS
- Authentication Policy
- Origin Authentication
- Authorization