

# Deep Dive into Istio Auth Policies

Lawrence Gadban / Solo.io



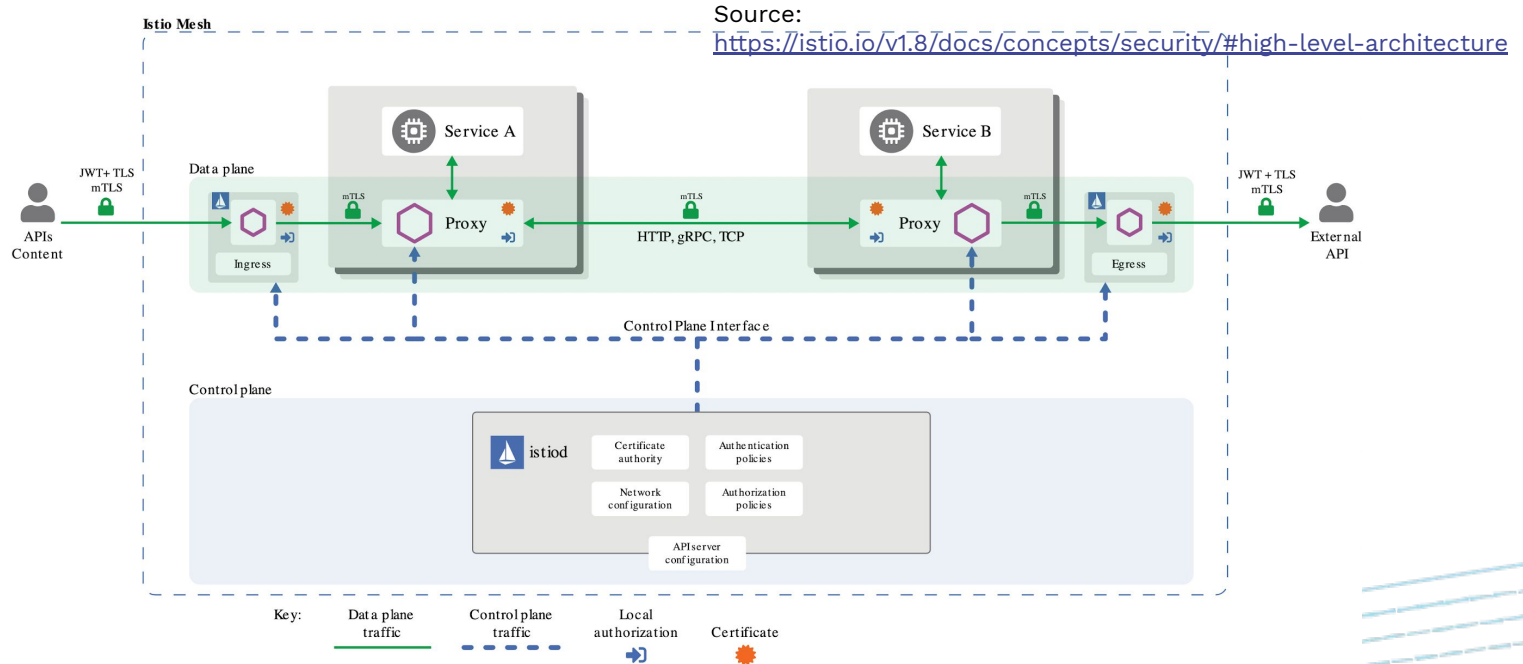
#IstioCon

# Intro

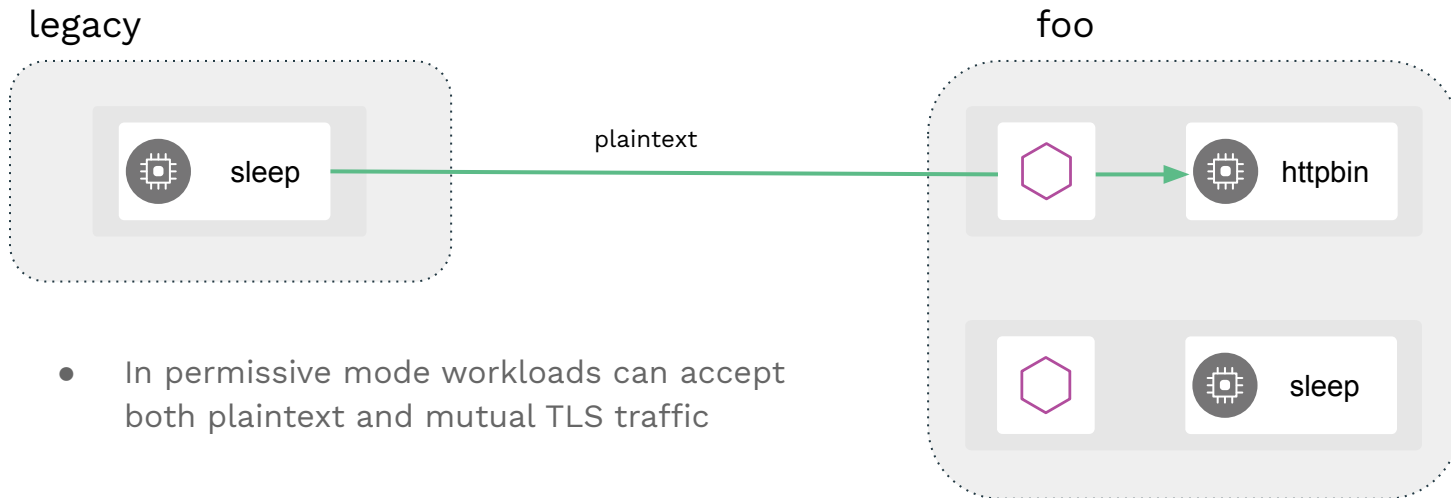
#IstioCon



# Istio Security Architecture



# Istio mTLS - Permissive Mode



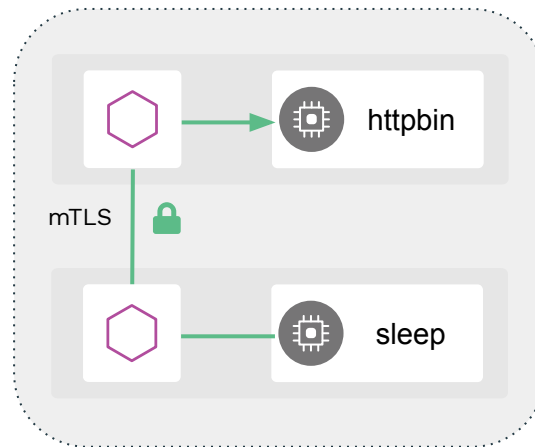
# Istio mTLS - Permissive Mode

legacy



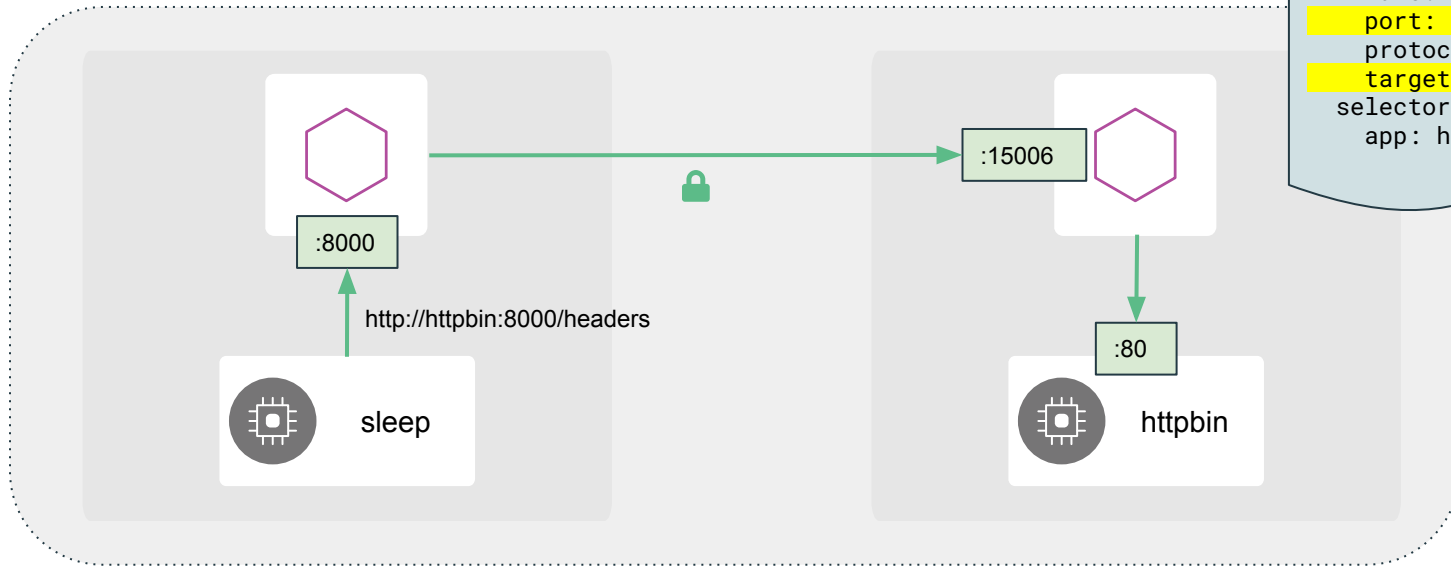
- Even in permissive mode, traffic between two workloads injected with sidecar proxies will use istio mutual TLS

foo



# Mutual TLS Traffic

foo



```
kind: Service
metadata:
  name: httpbin
  namespace: foo
spec:
  clusterIP: 10.97.186.224
  ports:
    - name: http
      port: 8000
      protocol: TCP
      targetPort: 80
  selector:
    app: httpbin
```



# Inbound Request

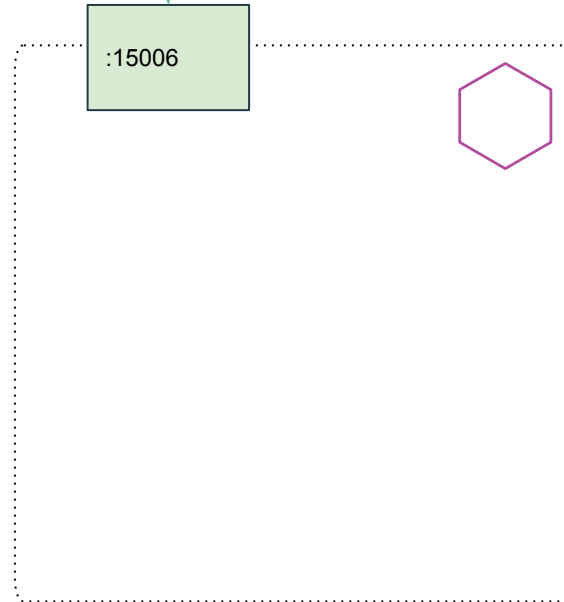
## Endpoints for httpbin Service

```
kind: Endpoints
metadata:
  labels:
    app: httpbin
    service: httpbin
  name: httpbin
  namespace: foo
subsets:
- addresses:
  - ip: 10.100.1.9
    targetRef:
      kind: Pod
      name: httpbin-75b587d94-fh8c1
      namespace: foo
  ports:
  - name: http
    port: 80
    protocol: TCP
```

Istio mTLS traffic **10.100.1.9:80** (ip addr of pod, dest. port of service)



http://httpbin:8000/headers



# Inbound Request

listener config

```
listener:  
  name: virtualInbound  
  address:  
    socketAddress:  
      address: 0.0.0.0  
      portValue: 15006  
  filterChains:  
  - filterChainMatch:  
      destinationPort: 80  
      transportProtocol: tls  
      applicationProtocols:  
        - istio  
        - istio-http/1.0  
        - istio-http/1.1  
        - istio-h2  
      transportSocket: {...}  
  filters:  
  - ...  
  - name: envoy.filters.network.http_connection_manager
```

iptables hijack

:15006

Istio mTLS traffic  
10.100.1.9:80 (ip addr of pod, dest. port of service)  
http://httpbin:8000/headers





# Inbound Request

listener config

```
listener:  
  name: virtualInbound  
  address:  
    socketAddress:  
      address: 0.0.0.0  
      portValue: 15006  
  filterChains:  
  - filterChainMatch:  
    destinationPort: 80  
    transportProtocol: tls  
    applicationProtocols:  
    - istio  
    - istio-http/1.0  
    - istio-http/1.1  
    - istio-h2  
  transportSocket: {...}  
  filters:  
  - ...  
  - name: envoy.filters.network.http_connection_manager
```



Istio mTLS traffic | 10.100.1.9:80 (ip addr of pod, dest. port of service)



http://httpbin:8000/headers


:15006

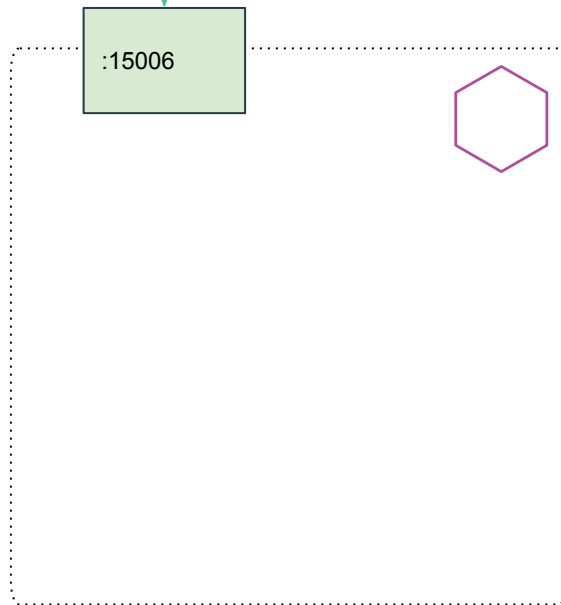


# Inbound Request

listener config

```
listener:  
  name: virtualInbound  
  address:  
    socketAddress:  
      address: 0.0.0.0  
      portValue: 15006  
  filterChains:  
  - filterChainMatch:  
    destinationPort: 80  
    transportProtocol: tls  
    applicationProtocols:  
    - istio  
    - istio-http/1.0  
    - istio-http/1.1  
    - istio-h2  
  transportSocket: {...}  
  filters:  
  - ...  
  - name: envoy.filters.network.http_connection_manager
```

Istio mTLS traffic  10.100.1.9:80 (ip addr of pod, dest. port of service)  
http://httpbin:8000/headers




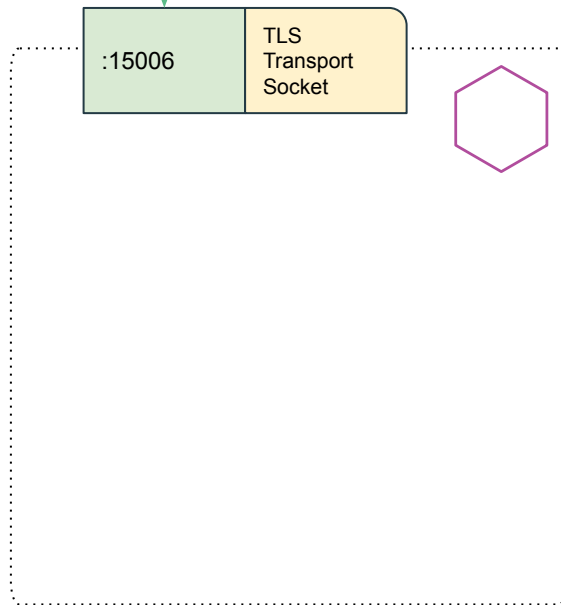
# Inbound Request

transport\_socket config

```
transport_socket:  
  common_tls_context:  
    tls_certificate_sds_secret_configs:  
      - {...}  
    combined_validation_context:  
      default_validation_context:  
        match_subject_alt_names:  
          - prefix: spiffe://cluster.local/  
            validation_context_sds_secret_config:  
              name: ROOTCA  
              sds_config: {...}  
      require_client_certificate: true
```

- Require client certificate (mutual TLS)
- Validate client certificate with ROOTCA
- Enforce valid SPIFFE client certificate
- Configure server certificate to be available via SDS (istio-agent)

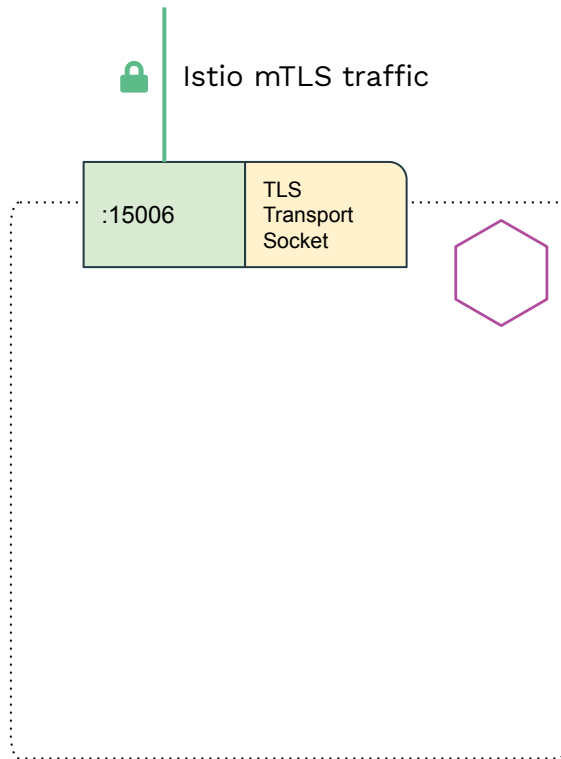
Istio mTLS traffic  10.100.1.9:80 (ip addr of pod, dest. port of service)  
http://httpbin:8000/headers



# Inbound Request

listener config

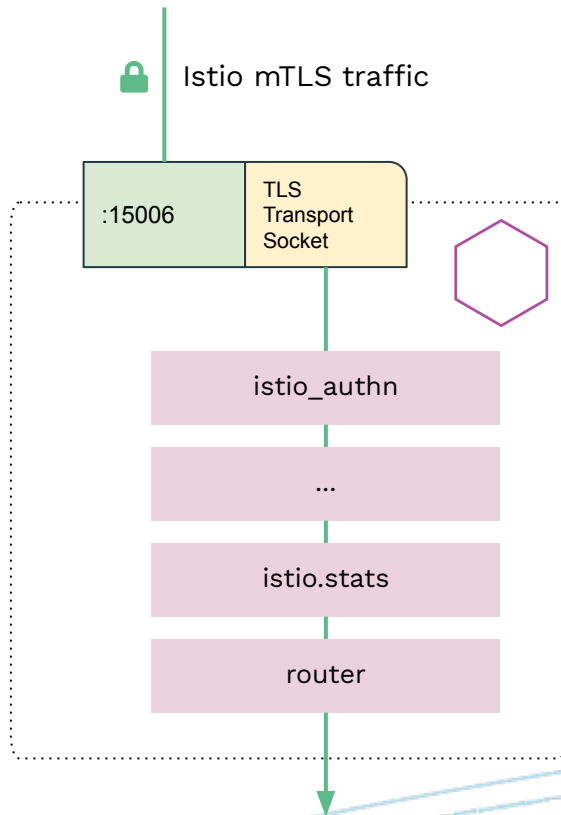
```
listener:  
  name: virtualInbound  
  address:  
    socketAddress:  
      address: 0.0.0.0  
      portValue: 15006  
  filterChains:  
  - filterChainMatch:  
    destinationPort: 80  
    transportProtocol: tls  
    applicationProtocols:  
    - istio  
    - istio-http/1.0  
    - istio-http/1.1  
    - istio-h2  
  transportSocket: {...}  
  filters:  
  - ...  
  - name: envoy.filters.network.http_connection_manager
```



# Inbound Request

network filters (specifically the HCM)

```
filters:  
- name: envoy.filters.network.http_connection_manager  
  typedConfig:  
    routeConfig: {}  
    httpFilters:  
    - ...  
    - name: istio_authn  
    - ...  
    - name: istio.stats  
    - name: envoy.filters.http.router
```



# Peer Authentication

#IstioCon



# PeerAuthentication Resource

- Controls how mutual TLS is enforced on selected workloads
- Policies can be scoped at different levels (mesh, namespace, workload, port)
- Most narrow scope takes precedence

```
apiVersion:  
"security.istio.io/v1beta1"  
kind: "PeerAuthentication"  
metadata:  
  name: "default"  
  namespace: "istio-system"  
spec:  
  mtls:  
    mode: STRICT
```



# PeerAuthentication Resource

- PERMISSIVE: Workloads accept both mutual TLS and plain text traffic. This mode is most useful during migrations when workloads without sidecar cannot use mutual TLS. Once workloads are migrated with sidecar injection, you should switch the mode to STRICT.
- STRICT: Workloads only accept mutual TLS traffic.
- DISABLE: Mutual TLS is disabled. From a security perspective, you shouldn't use this mode unless you provide your own security solution.
- UNSET: Inherit from parent, if has one. Otherwise treated as PERMISSIVE.

```
apiVersion:  
"security.istio.io/v1beta1"  
kind: "PeerAuthentication"  
metadata:  
  name: "default"  
  namespace: "istio-system"  
spec:  
  mtls:  
    mode: STRICT
```

Source: <https://istio.io/v1.8/docs/concepts/security/#peer-authentication>

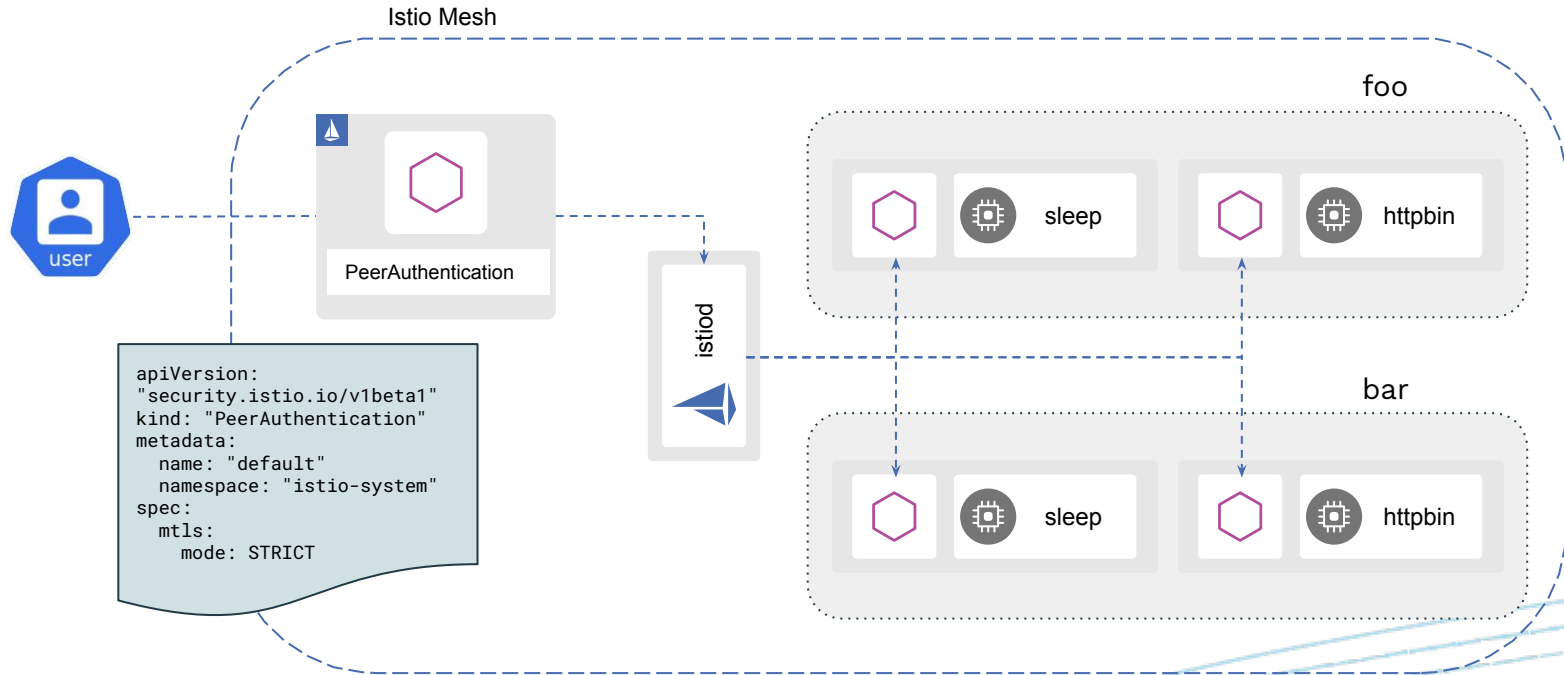
Source:

[https://istio.io/v1.8/docs/reference/config/security/peer\\_authentication/#PeerAuthentication-MutualTLS-Mode](https://istio.io/v1.8/docs/reference/config/security/peer_authentication/#PeerAuthentication-MutualTLS-Mode)

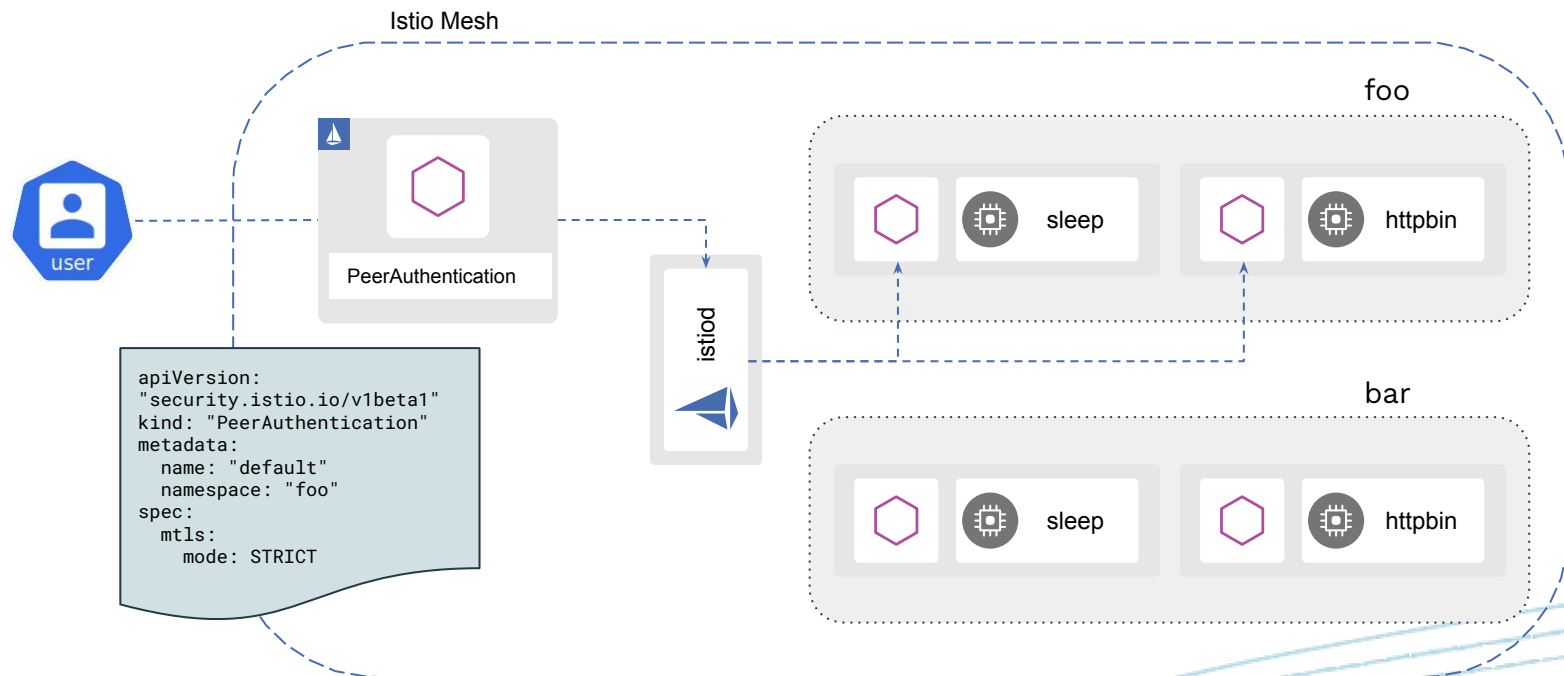




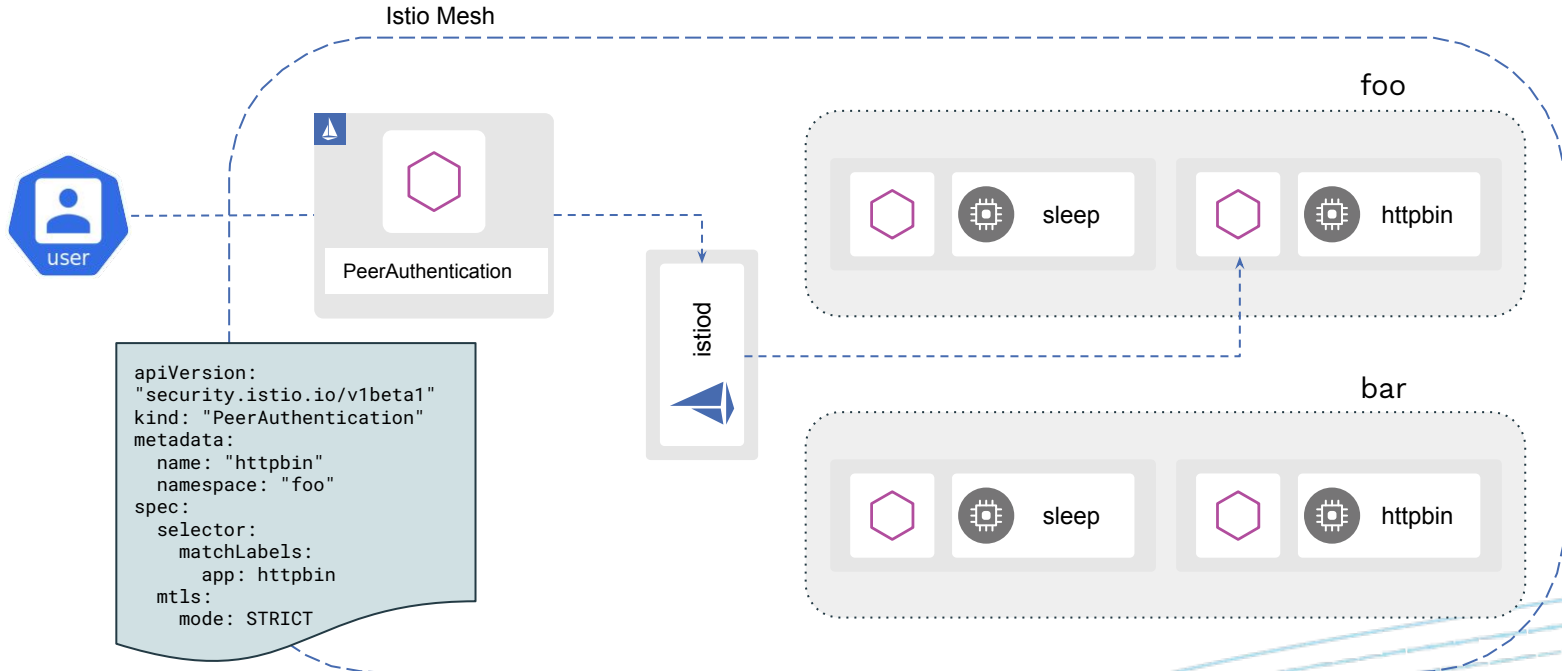
# Mesh scope



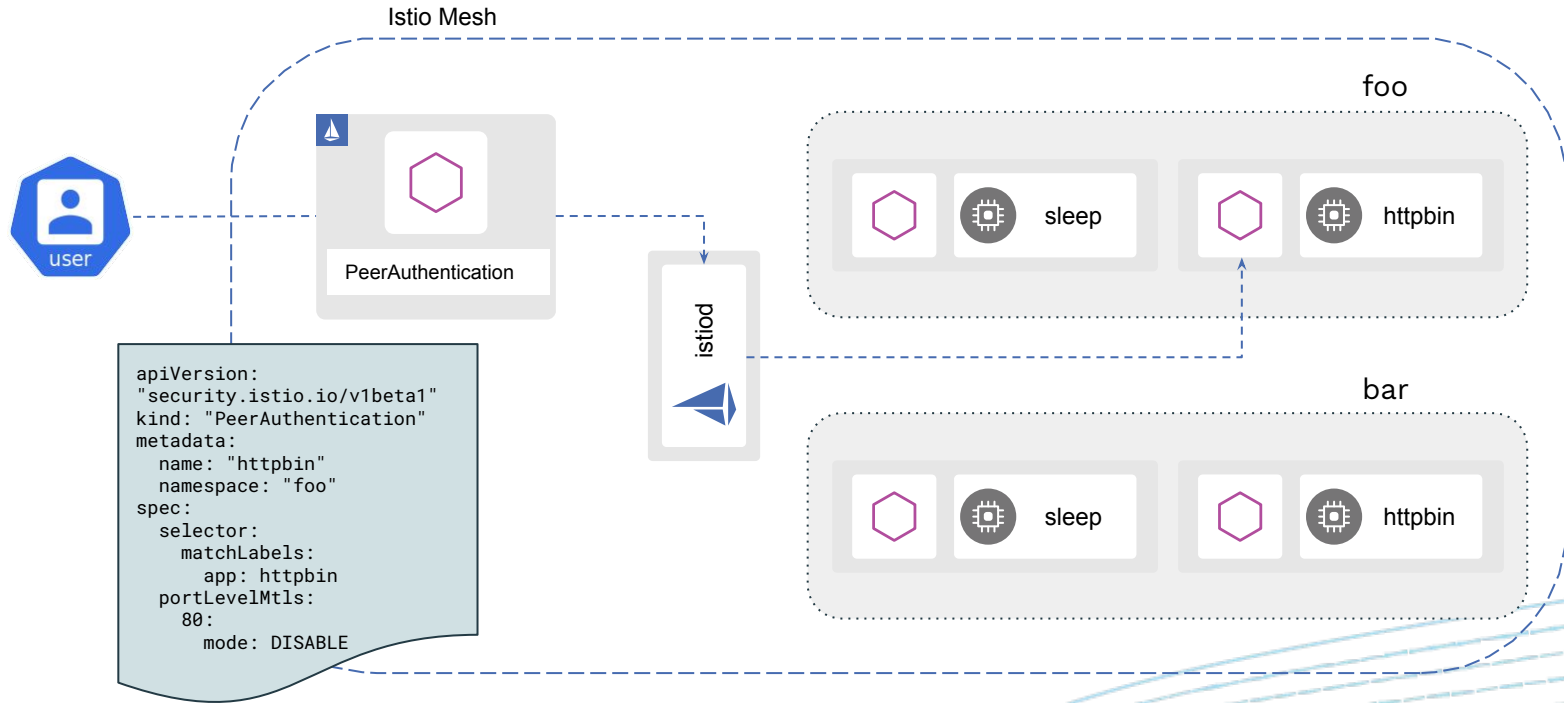
# Namespace scope



# Workload scope



# Port-level scope



# Permissive Mode

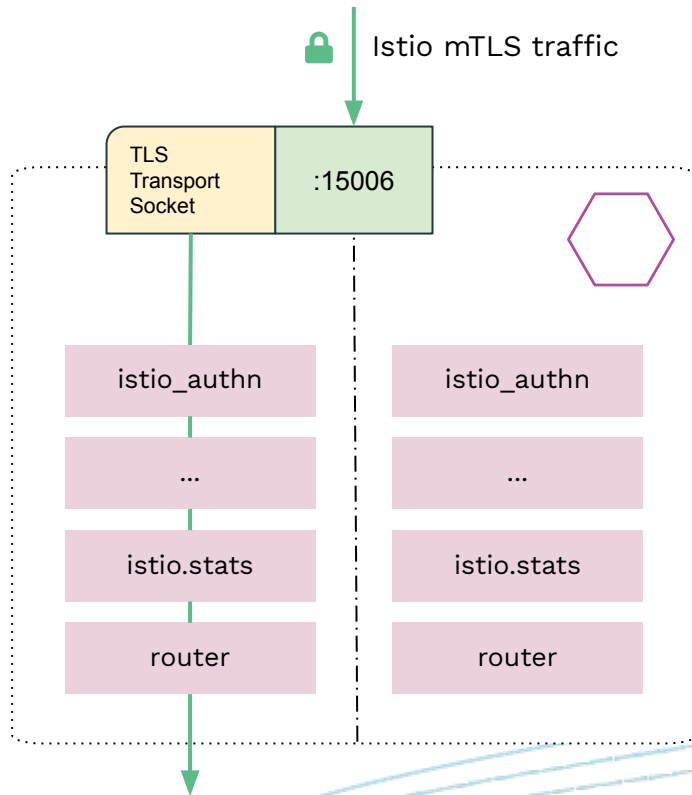
mTLS match

```
"filter_chain_match": {  
  "destination_port": 80,  
  "transport_protocol": "tls",  
  "application_protocols": [  
    "istio",  
    "istio-http/1.0",  
    "istio-http/1.1",  
    "istio-h2"  
  ]  
}
```



plaintext match

```
filter_chain_match: {  
  destination_port: 80,  
  transport_protocol: "raw_buffer"  
}
```



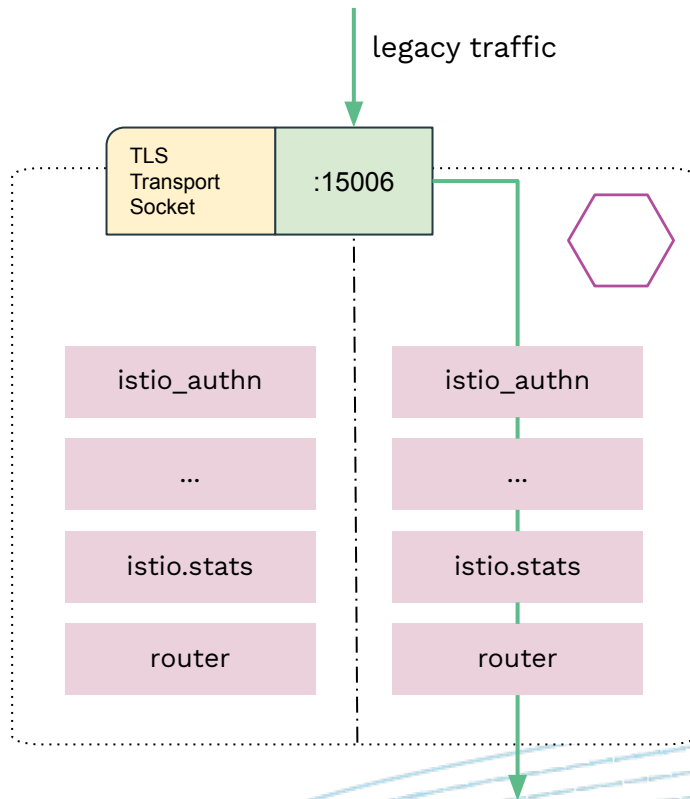
# Permissive Mode

mTLS match

```
"filter_chain_match": {  
  "destination_port": 80,  
  "transport_protocol": "tls",  
  "application_protocols": [  
    "istio",  
    "istio-http/1.0",  
    "istio-http/1.1",  
    "istio-h2"  
  ]  
}
```

plaintext match

```
filter_chain_match: {  
  destination_port: 80,  
  transport_protocol: "raw_buffer"  
}
```



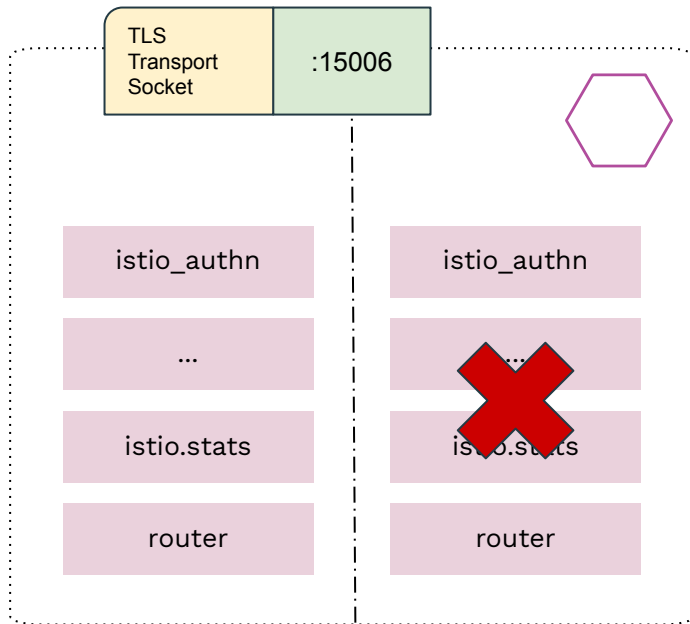
# Strict Mode

TLS match

```
"filter_chain_match": {  
  "destination_port": 80,  
  "transport_protocol": "tls"  
}
```

plaintext match

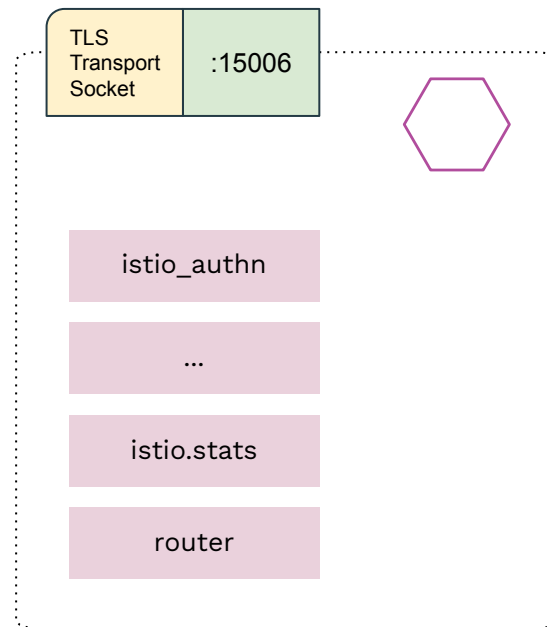
```
filter_chain_match: {  
  destination_port: 80,  
  transport_protocol: "raw_buffer"  
}
```



# Strict Mode

TLS match

```
"filter_chain_match": {  
  "destination_port": 80,  
  "transport_protocol": "tls"  
}
```





# Strict Mode

TLS match

```
"filter_chain_match": {  
  "destination_port": 80,  
  "transport_protocol": "tls"  
}
```

**X**  
No match!

legacy traffic

TLS  
Transport  
Socket  
:15006

**% curl http://httpbin.foo:8000/headers -s  
command terminated with exit code 56**

istio\_authn

```
2021-02-19T17:00:14.584216Z    debug    envoy filter    original_dst: New connection accepted  
2021-02-19T17:00:14.584254Z    debug    envoy filter    tls inspector: new connection accepted  
2021-02-19T17:00:14.584283Z    debug    envoy conn_handler  closing connection: no matching filter chain found
```

istio.stats

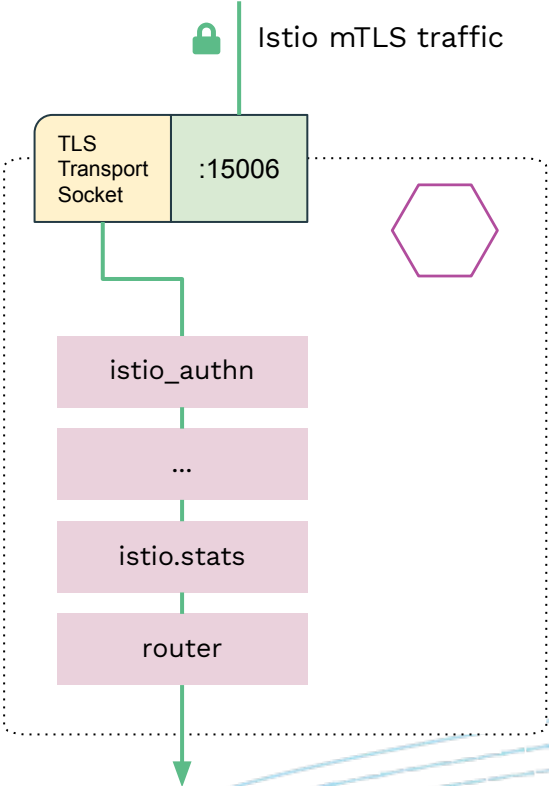
router



# Strict Mode

TLS match

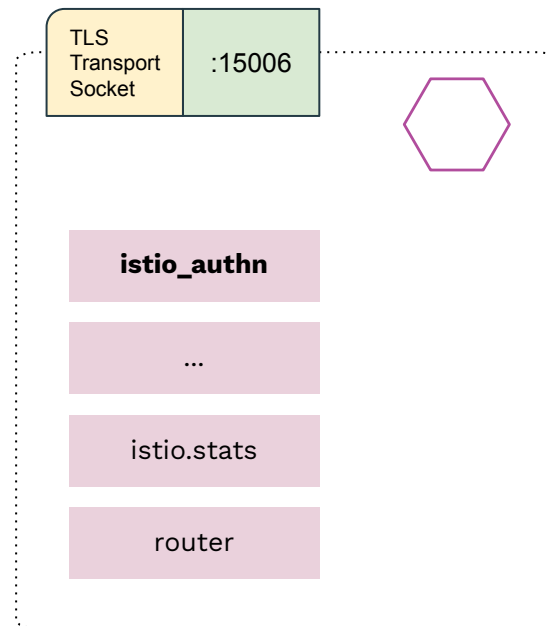
```
"filter_chain_match": {  
  "destination_port": 80,  
  "transport_protocol": "tls"  
}
```



# Permissive Mode

Istio\_authn filter config

```
name: istio_authn
policy:
  peers:
    - mtls:
        mode: PERMISSIVE
skip_validate_trust_domain: true
```

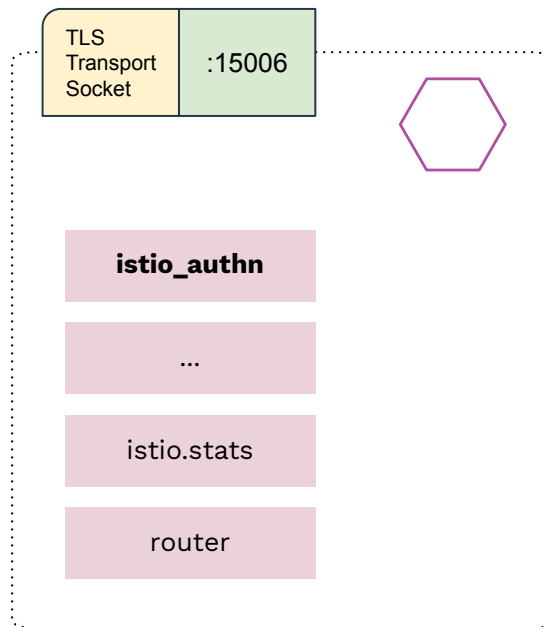


# Strict Mode

Istio\_authn filter config

```
name: istio_authn
policy:
  peers:
    - mtls: {}
skip_validate_trust_domain: true
```

- Validates the connection uses mTLS and the peer contains a valid SPIFFE cert
- Saves the peer auth data for use in later filters
- Does not validate trust domain, AuthZ policies can enforce



# Strict Mode

proxy debug logs -- istio\_authn filter

```
2021-02-23T05:32:36.835994Z    debug    envoy filter    [C213429] validateX509 mode STRICT: ssl=true, has_user=true
2021-02-23T05:32:36.836043Z    debug    envoy filter    [C213429] trust domain validation skipped
2021-02-23T05:32:36.836085Z    debug    envoy filter    Set peer from X509: cluster.local/ns/foo/sa/sleep
2021-02-23T05:32:36.836104Z    debug    envoy filter    Set principal from peer: cluster.local/ns/foo/sa/sleep
2021-02-23T05:32:36.836180Z    debug    envoy filter    Origin authenticator succeeded
2021-02-23T05:32:36.836316Z    debug    envoy filter    Saved Dynamic Metadata:
fields {
  key: "request.auth.principal"
  value {
    string_value: "cluster.local/ns/foo/sa/sleep"
  }
}
fields {
  key: "source.namespace"
  value {
    string_value: "foo"
  }
}
fields {
  key: "source.principal"
  value {
    string_value: "cluster.local/ns/foo/sa/sleep"
  }
}
```

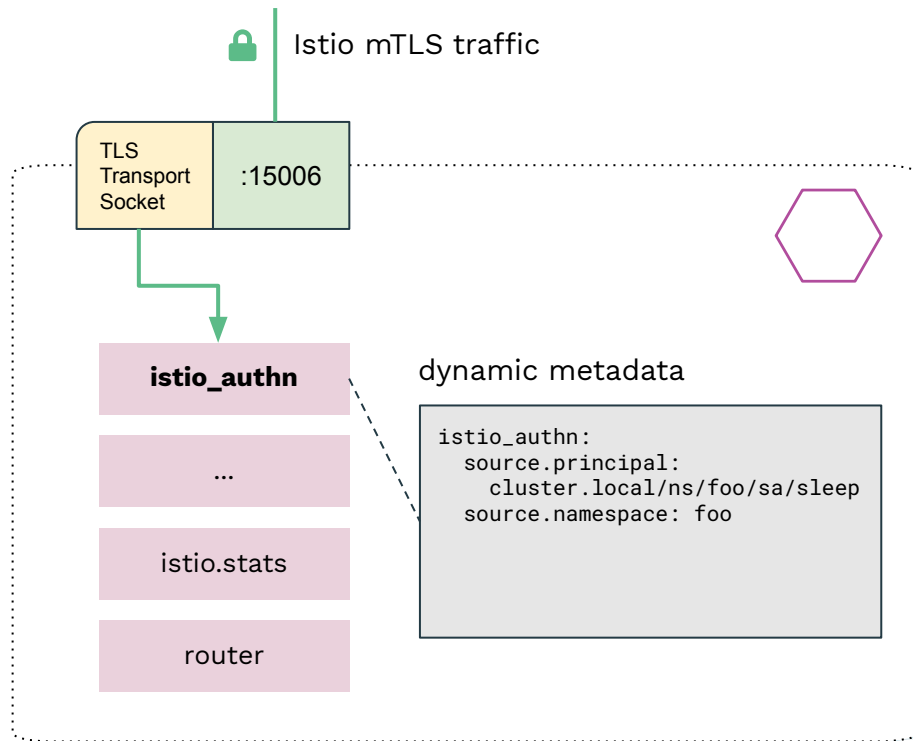


# Istio AuthN Filter

Istio\_authn filter config

```
name: istio_authn
policy:
  peers:
    - mtls: {}
  skip_validate_trust_domain: true
```

- Validates the connection uses mTLS and the peer contains a valid SPIFFE cert
- **Saves the peer auth data for use in later filters**
- Does not validate trust domain, AuthZ policies can enforce



# Request Authentication

#IstioCon



# RequestAuthentication Resource

Request authentication policies specify the values needed to validate a JSON Web Token (JWT). These values include, among others, the following:

- The location of the token in the request
- The issuer or the request
- The public JSON Web Key Set (JWKS)

Istio checks the presented token, if presented against the rules in the request authentication policy, and rejects requests with invalid tokens. When requests carry no token, they are accepted by default. To reject requests without tokens, provide authorization rules that specify the restrictions for specific operations, for example paths or actions.

Source: <https://istio.io/v1.8/docs/concepts/security/#request-authentication>

```
apiVersion: "security.istio.io/v1beta1"
kind: "RequestAuthentication"
metadata:
  name: "foo-jwt"
  namespace: "foo"
spec:
  selector:
    matchLabels:
      app: httpbin
  jwtRules:
    - issuer: "foo.io"
      jwksUri: "https://foo.io/jwks.json"
```





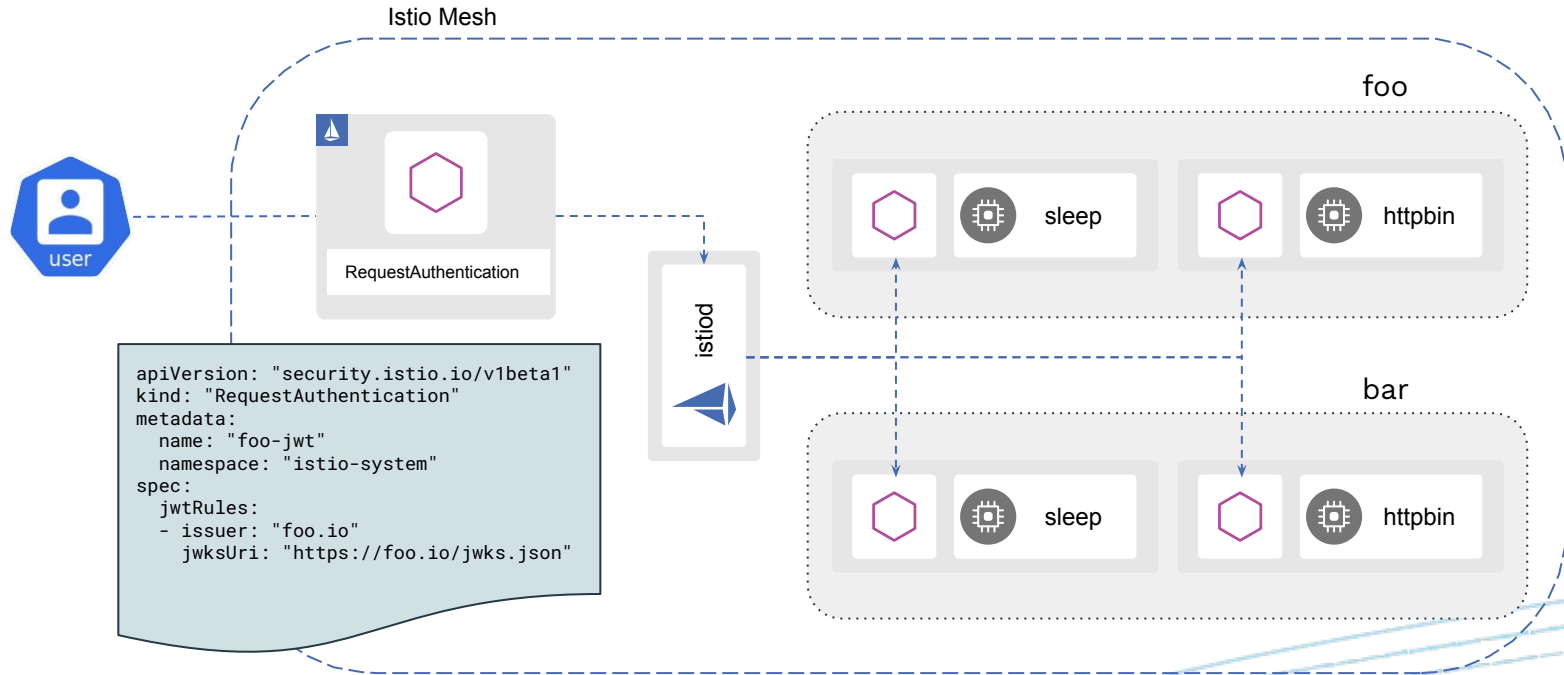
# RequestAuthentication Resource

- Policies can be scoped at different levels (mesh, namespace, workload)
- Multiple policies can be applied to a single workload, rules will be aggregated allowing for multiple providers

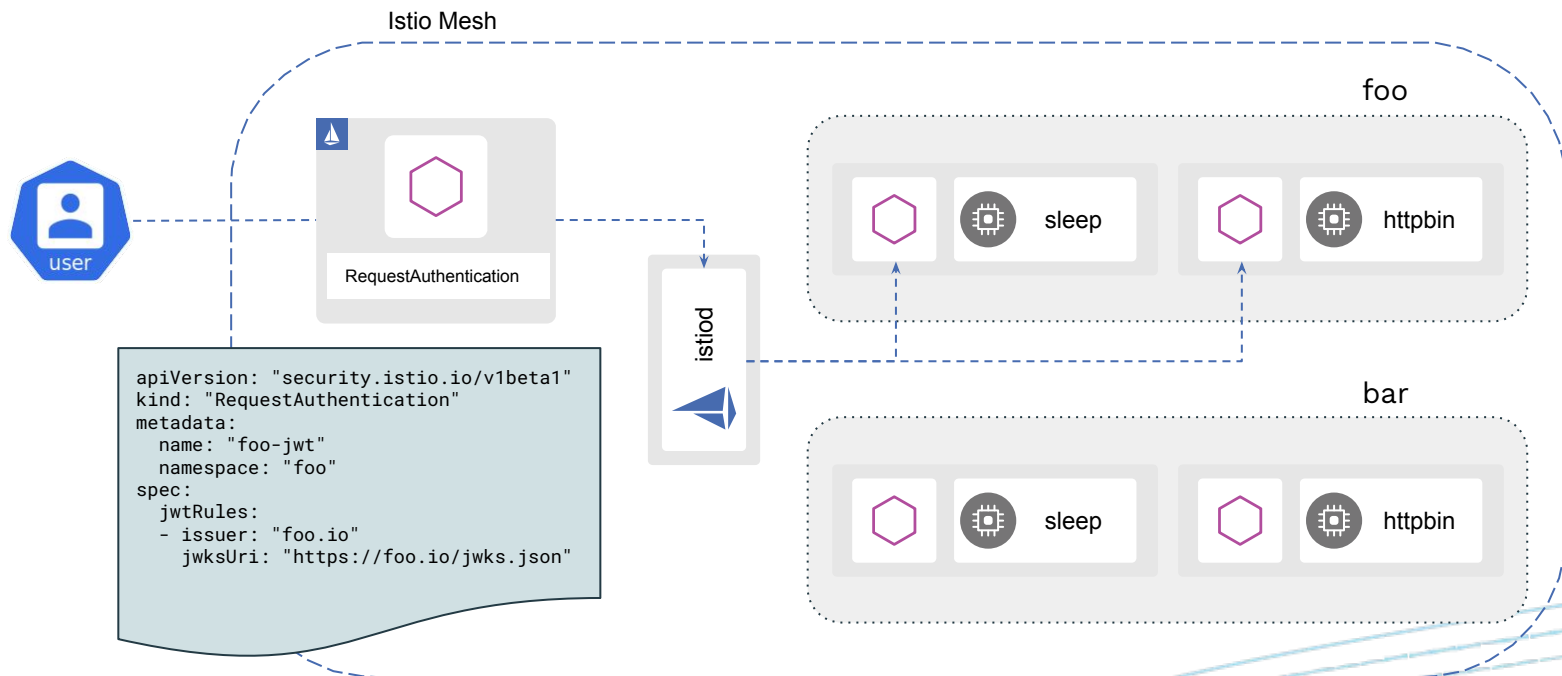
```
apiVersion: "security.istio.io/v1beta1"
kind: "RequestAuthentication"
metadata:
  name: "foo-jwt"
  namespace: "foo"
spec:
  selector:
    matchLabels:
      app: httpbin
  jwtRules:
    - issuer: "foo.io"
      jwksUri: "https://foo.io/jwks.json"
```



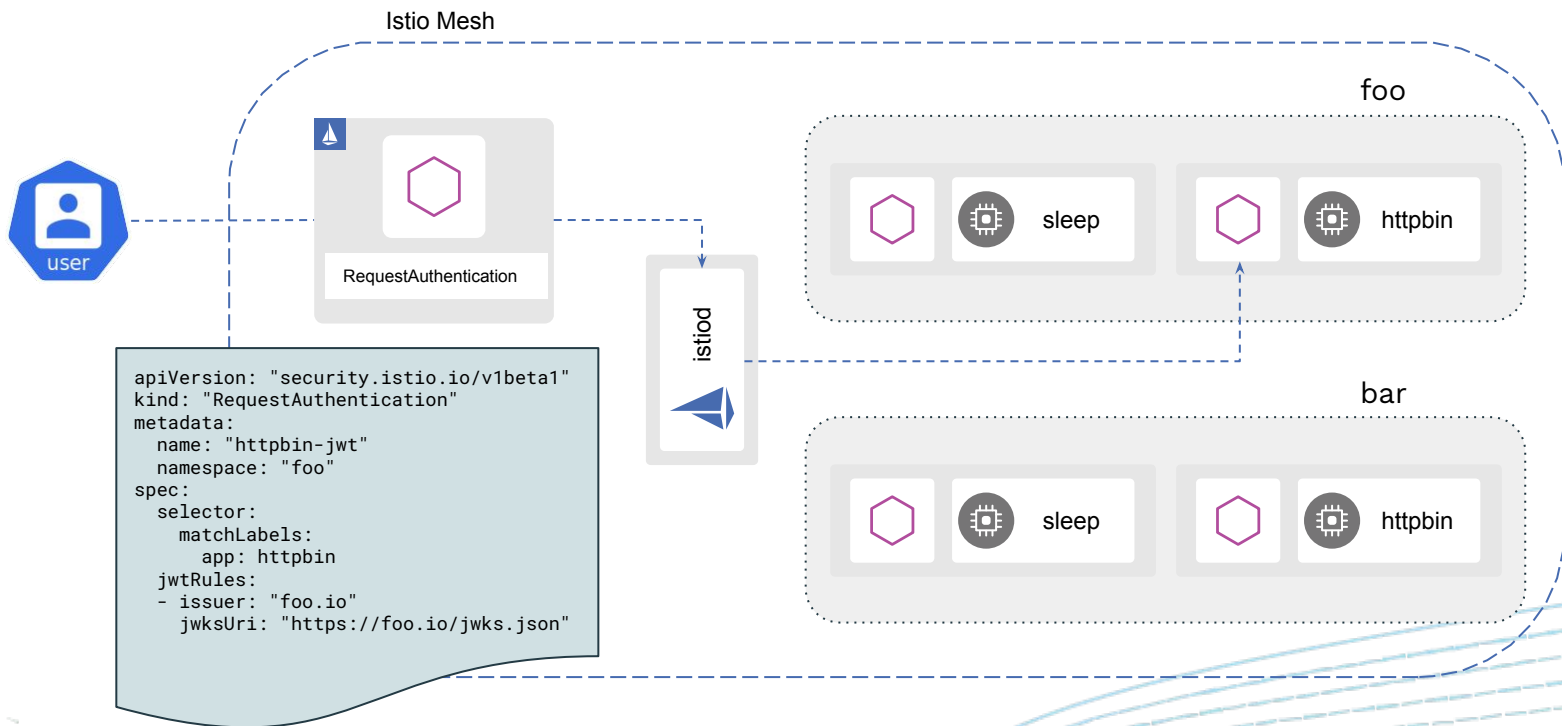
# Mesh scope



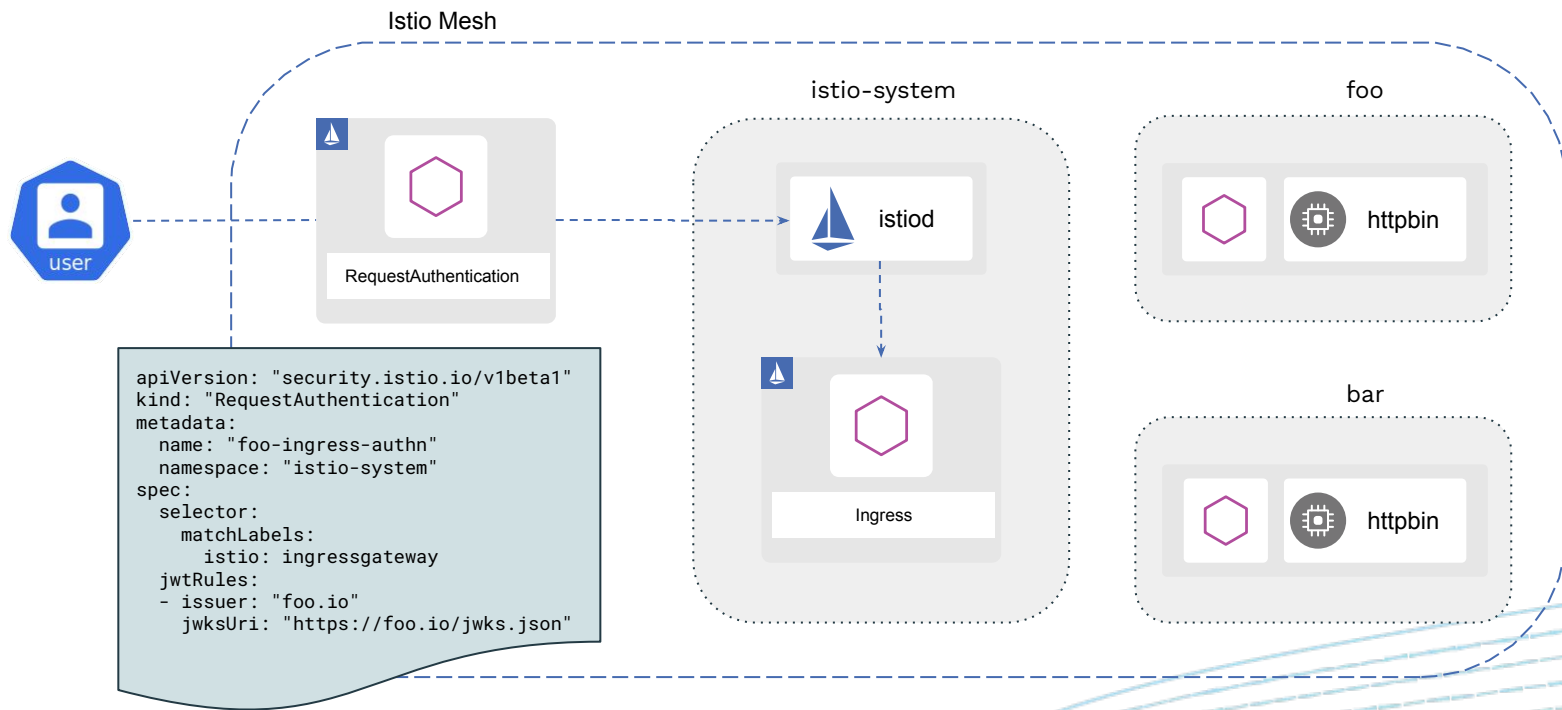
# Namespace scope



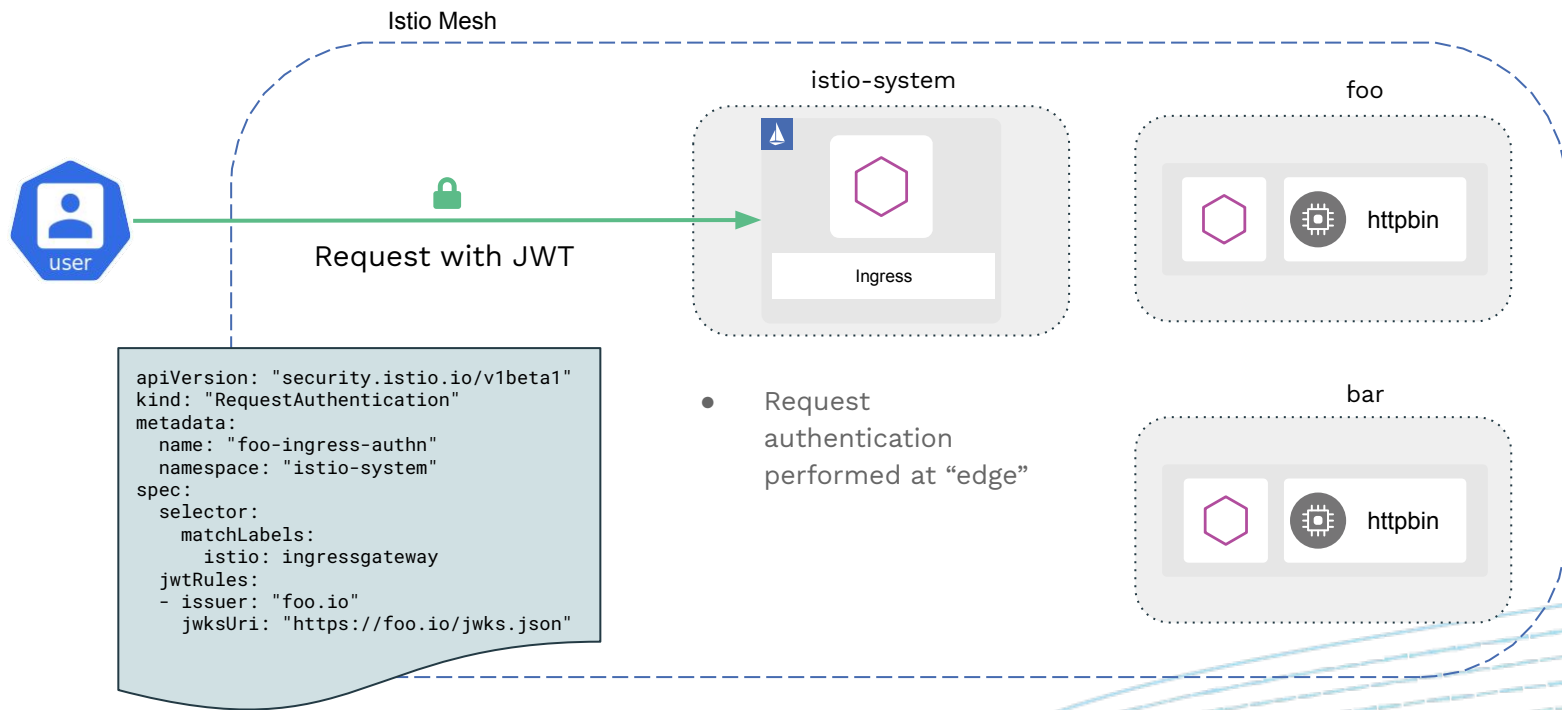
# Workload scope



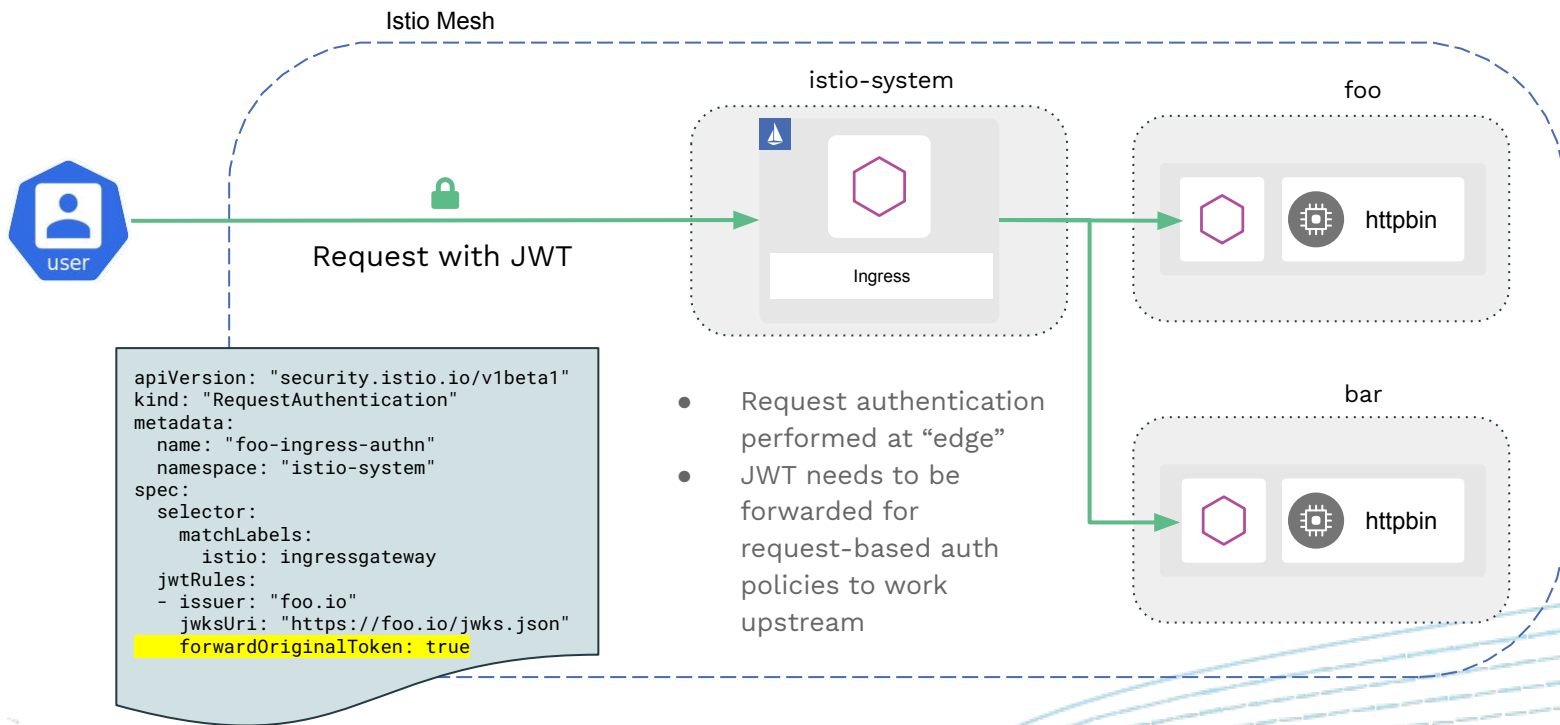
# Request AuthN @ Ingress



# Request AuthN @ Ingress



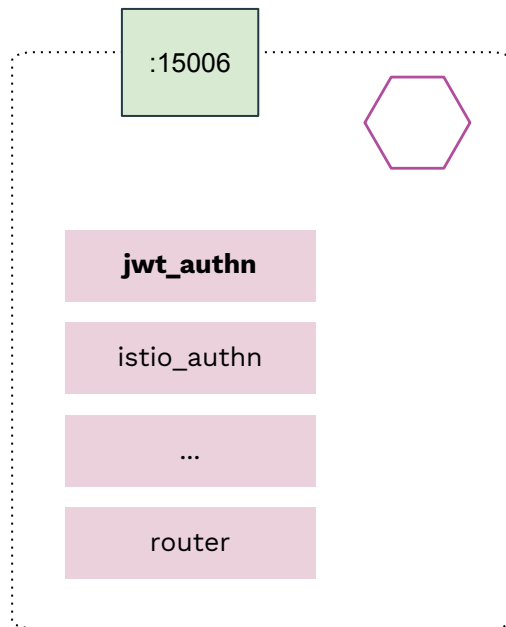
# Request AuthN @ Ingress



# JWT Filter

```
name: envoy.filters.http.jwt_authn
providers:
  origins-0:
    issuer: foo.global-corp.io
    localJwks:
      inlineString: ...
    payloadInMetadata: foo.global-corp.io
rules:
- match:
  prefix: "/"
  requires:
    requiresAny:
      requirements:
      - providerName: origins-0
      - allowMissing: {}
```

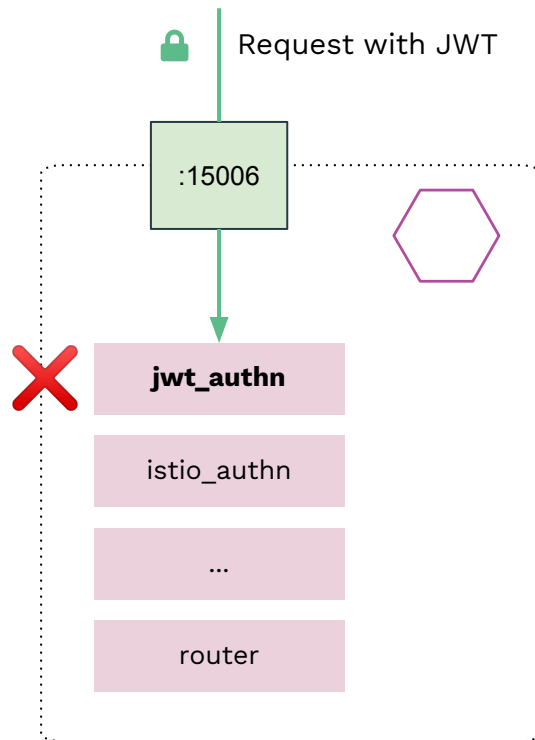
- Envoy JWT filter performs enforcement
- If JWT is present, it must be valid
- Allows requests with no JWT present to pass through (i.e. authorization must be performed to deny requests with no JWT)





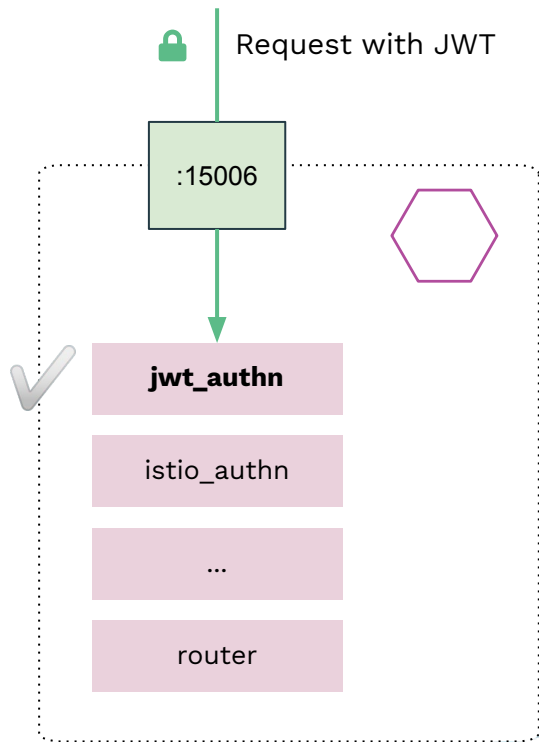
# JWT Filter

- JWT issuer doesn't match
- Signature can't be verified with public key
- JWT is expired
- JWT doesn't contain a valid audience (if defined)



# JWT Filter

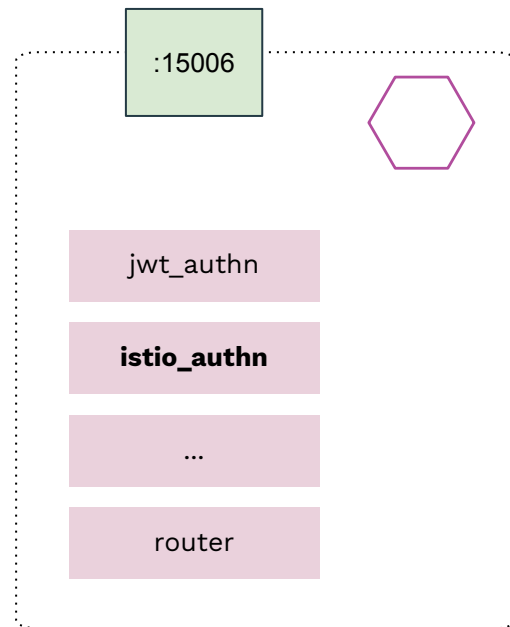
- Valid JWT
- Request with no JWT (!)



# Istio AuthN Filter

```
name: istio_authn
policy:
  peers:
    - mtls: {}
  origins:
    - jwt:
        issuer: foo.global-corp.io
      originIsOptional: true
      principalBinding: USE_ORIGIN
      skipValidateTrustDomain: true
```

- Since “originIsOptional”, not much enforcement in the istio authn filter
- Bind request principal from relevant JWT claims (i.e. ‘iss’ and ‘sub’ claims)
- Saves authentication results (i.e. JWT claims) to Envoy dynamic metadata for use in later filters (e.g. authorization)



# Istio AuthN Filter

proxy debug logs -- istio\_authn filter

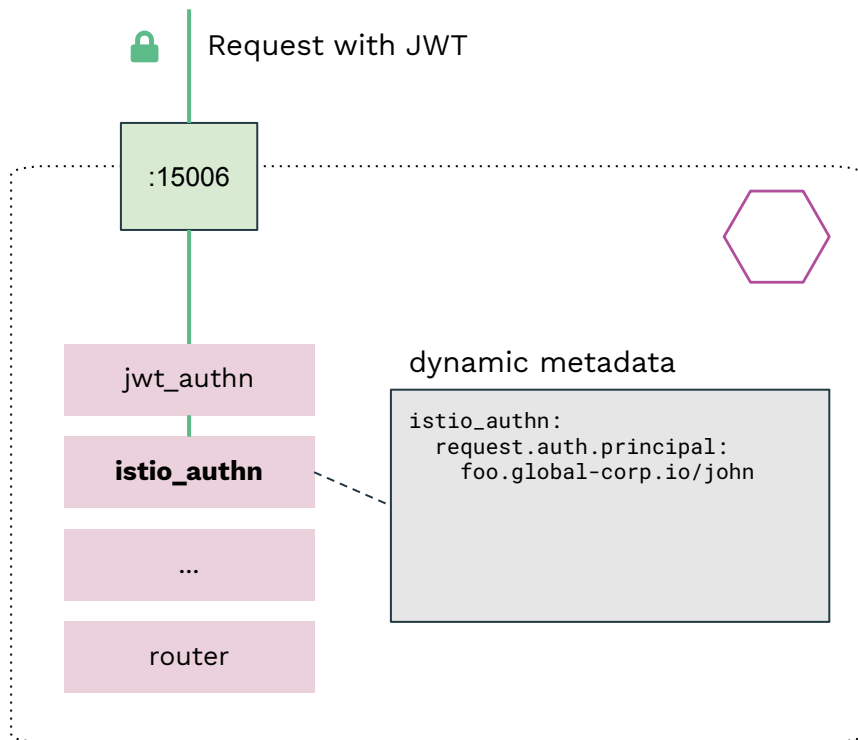
```
2021-02-23T15:32:52.137638Z  debug  envoy filter  ProcessJwtPayload: json object is
{"admin":true,"iat":1516239022,"iss":"foo.global-corp.io","name":"John Doe","sub":"john"}
2021-02-23T15:32:52.137701Z  debug  envoy filter  JWT validation succeeded
2021-02-23T15:32:52.137715Z  debug  envoy filter  Set principal from origin: foo.global-corp.io/john
2021-02-23T15:32:52.137719Z  debug  envoy filter  Origin authenticator succeeded
2021-02-23T15:32:52.137922Z  debug  envoy filter  Saved Dynamic Metadata:
fields {
  key: "request.auth.claims"
  value {
    ... full JWT claims here ...
  }
}
fields {
  key: "request.auth.principal"
  value {
    string_value: "foo.global-corp.io/john"
  }
}
fields {
  key: "request.auth.raw_claims"
  value {
    string_value: "{\"iss\":\"foo.global-corp.io\",\"admin\":true,\"name\":\"John
Doe\",\"sub\":\"john\",\"iat\":1516239022}"
  }
}
```



# Istio AuthN Filter

```
name: istio_authn
policy:
  peers:
  - mtls: {}
  origins:
  - jwt:
      issuer: foo.global-corp.io
    originIsOptional: true
    principalBinding: USE_ORIGIN
  skipValidateTrustDomain: true
```

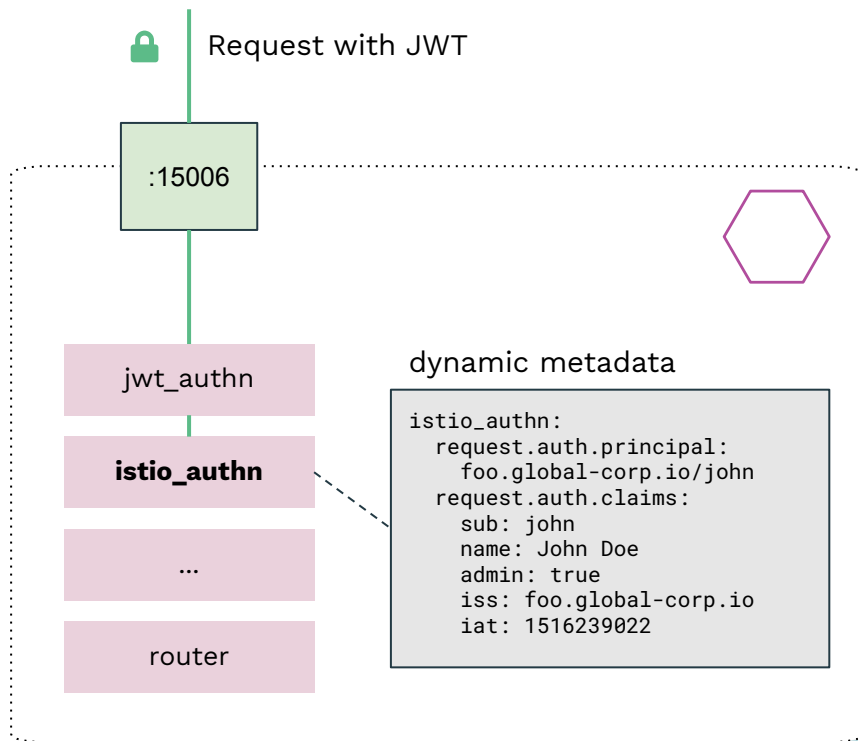
- Since “originIsOptional”, not much enforcement in the istio authn filter
- **Bind request principal from relevant JWT claims (i.e. ‘iss’ and ‘sub’ claims)**
- Saves authentication results (i.e. JWT claims) to Envoy dynamic metadata for use in later filters (e.g. authorization)



# Istio AuthN Filter

```
name: istio_authn
policy:
  peers:
    - mtls: {}
  origins:
    - jwt:
        issuer: foo.global-corp.io
      originIsOptional: true
      principalBinding: USE_ORIGIN
      skipValidateTrustDomain: true
```

- Since “originIsOptional”, not much enforcement in the istio authn filter
- Bind request principal from relevant JWT claims (i.e. ‘iss’ and ‘sub’ claims)
- **Saves authentication results (i.e. JWT claims) to Envoy dynamic metadata for use in later filters (e.g. authorization)**



# Authorization Policies

#IstioCon



# AuthorizationPolicy Resource

Istio's authorization features provide mesh-, namespace-, and workload-wide access control for your workloads in the mesh.

- The selector field specifies the target of the policy
- The action field specifies whether to allow or deny the request
- The rules specify when to trigger the action
  - The from field in the rules specifies the sources of the request
  - The to field in the rules specifies the operations of the request
  - The when field specifies the conditions needed to apply the rule

```
apiVersion: "security.istio.io/v1beta1"
kind: "AuthorizationPolicy"
metadata:
  name: "httpbin-viewer"
  namespace: foo
spec:
  selector:
    matchLabels:
      app: httpbin
  action: ALLOW
  rules:
    - from:
      - source:
          requestPrincipals:
            - "foo.global-corp.io/john"
      to:
        - operation:
            methods:
              - "GET"
```

Source: <https://istio.io/v1.8/docs/concepts/security/#authorization>

Source: <https://istio.io/v1.8/docs/concepts/security/#authorization-policies>





# AuthorizationPolicy Resource

- Implicit enablement; when no authz policies present, all traffic allowed
- One or more ALLOW present, traffic denied unless explicitly authorized via the ALLOW policies
- DENY policies evaluated first
- Multiple policies will be aggregated

```
apiVersion: "security.istio.io/v1beta1"
kind: "AuthorizationPolicy"
metadata:
  name: "httpbin-viewer"
  namespace: foo
spec:
  selector:
    matchLabels:
      app: httpbin
  action: ALLOW
  rules:
    - from:
      - source:
          requestPrincipals:
            - "foo.global-corp.io/john"
      to:
        - operation:
            methods:
              - "GET"
```



# AuthorizationPolicy Resource

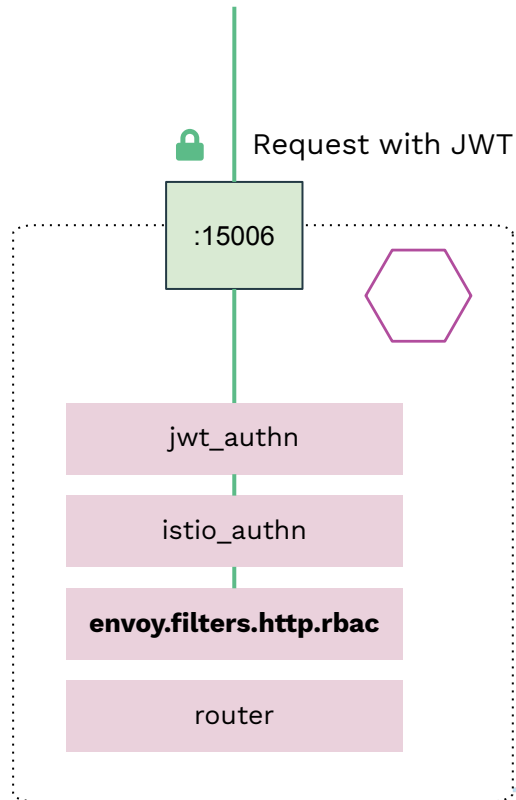
- Consider using a catch-all DENY rule to simplify mental model and not authorize requests unintentionally (either due to authorization policy changes or unauthenticated requests)

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: deny-all
  namespace: istio-system
spec:
  {}
```



# Envoy RBAC Filter

```
name: envoy.filters.http.rbac
rules:
  policies:
    ns[foo]-policy[httpbin-admin]-rule[0]:
      permissions:
        - andRules:
            rules:
              - orRules:
                  rules:
                    - header:
                        name: ":method"
                        exactMatch: GET
      principals:
        - andIds:
            ids:
              - orIds:
                  ids:
                    - metadata:
                        filter: istio_authn
                        path:
                          - key: request.auth.principal
                          value:
                            stringMatch:
                              exact: foo.global-corp.io/john
```



# Envoy RBAC Filter

```
% curl http://httpbin:8000/headers -s -H "authorization: Bearer ${FOO_TOKEN}"
```

```
name: envoy.filters.http.rbac
rules:
  policies:
    ns[foo]-policy[httpbin-admin]-rule[0]:
      permissions:
        - andRules:
            rules:
              - orRules:
                  rules:
                    - header:
                        name: ":method"
                        exactMatch: GET
            principals:
              - andIds:
                  ids:
                    - orIds:
                        ids:
                          - metadata:
                              filter: istio_authn
                              path:
                                - key: request.auth.principal
                                value:
                                  stringMatch:
                                    exact: foo.global-corp.io/john
```



Request with JWT

:15006



jwt\_authn

istio\_authn

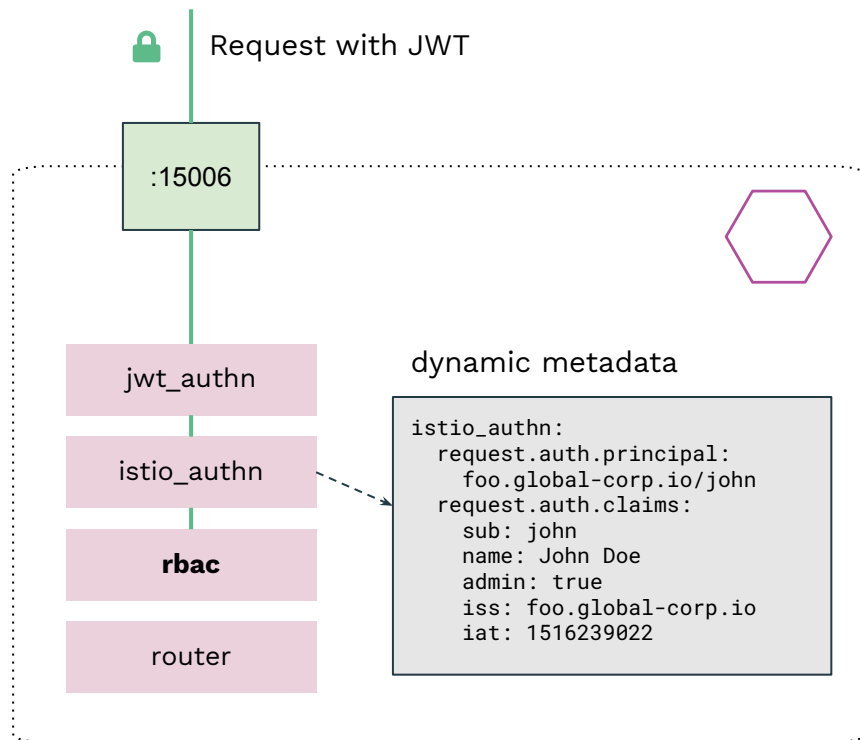
**envoy.filters.http.rbac**

router



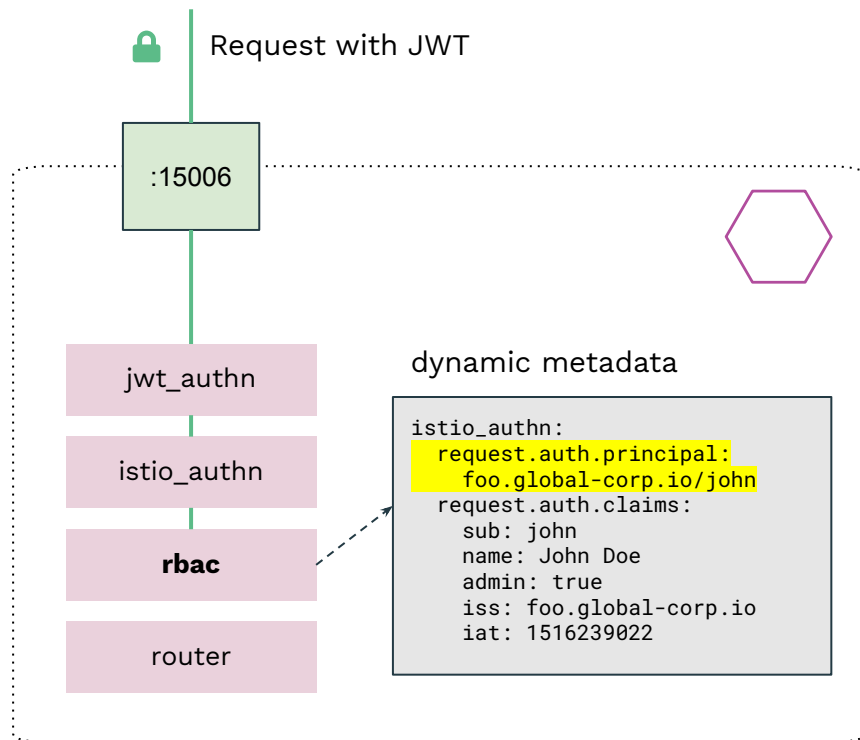
# Envoy RBAC Filter

```
name: envoy.filters.http.rbac
rules:
  policies:
    ns[foo]-policy[httpbin-admin]-rule[0]:
      permissions:
        - andRules:
            rules:
              - orRules:
                  rules:
                    - header:
                        name: ":method"
                        exactMatch: GET
      principals:
        - andIds:
            ids:
              - orIds:
                  ids:
                    - metadata:
                        filter: istio_authn
                        path:
                        - key: request.auth.principal
                        value:
                            stringMatch:
                                exact: foo.global-corp.io/john
```



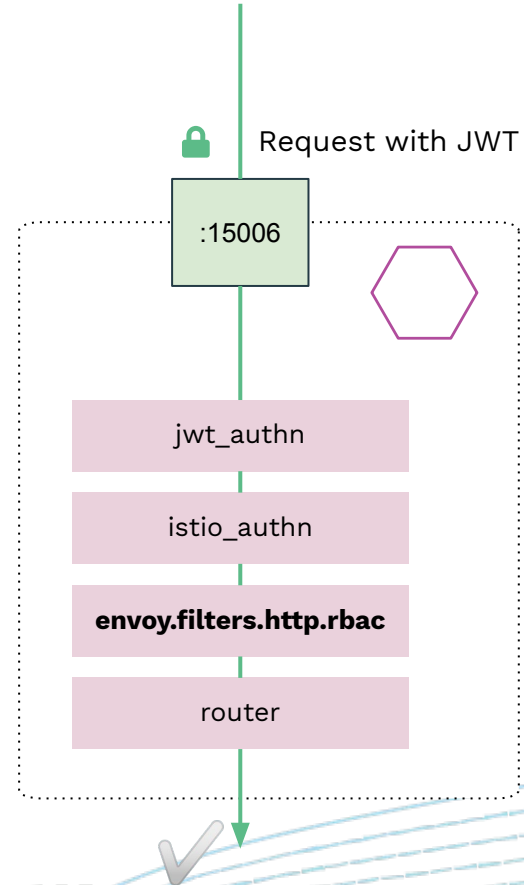
# Envoy RBAC Filter

```
name: envoy.filters.http.rbac
rules:
  policies:
    ns[foo]-policy[httpbin-admin]-rule[0]:
      permissions:
        - andRules:
            rules:
              - orRules:
                  rules:
                    - header:
                        name: ":method"
                        exactMatch: GET
      principals:
        - andIds:
            ids:
              - orIds:
                  ids:
                    - metadata:
                        filter: istio_authn
                        path:
                        - key: request.auth.principal
                        value:
                            stringMatch:
                                exact: foo.global-corp.io/john
```



# Envoy RBAC Filter

```
name: envoy.filters.http.rbac
rules:
  policies:
    ns[foo]-policy[httpbin-admin]-rule[0]:
      permissions:
        - andRules:
            rules:
              - orRules:
                  rules:
                    - header:
                        name: ":method"
                        exactMatch: GET
      principals:
        - andIds:
            ids:
              - orIds:
                  ids:
                    - metadata:
                        filter: istio_authn
                        path:
                          - key: request.auth.principal
                          value:
                            stringMatch:
                              exact: foo.global-corp.io/john
```





**SAVE THE DATE!**

March 23-25, 2021

**solo**  **con**

The mascot is a blue, rounded character with a single antenna on its head, a smiling mouth, and small arms and legs. It is positioned between the words 'solo' and 'con'.

**DIGITAL EXPERIENCE**

[solo.io/solocon/](https://solo.io/solocon/)



# Thank you!

[lawrence.gadban@solo.io](mailto:lawrence.gadban@solo.io)

icons/diagram attribution

Istio diagram creation guide: <https://istio.io/v1.8/about/contribute/diagrams/>

Kubernetes community icons: <https://github.com/kubernetes/community/tree/master/icons>

#IstioCon

