Istio AuthN filter revisit

Author(s): rei@tetrate.io, lizan@tetrate.io
Work-Group: Security
Short self link:
Contributors:
Reviewers: @liminw, @yonggangl, @yangminzhu,

Status: WIP | **In Review** | Approved | Obsolete
Created: 2020-08-12
Review By: YYYY-MON-DD
Release Version: 1.8
Approvers: liminw[ ], yonggongl[ ]

Objective

Istio AuthN filter has old codes. Currently, we have proxy-wasm extensibility, and upstream RBAC and JWT AuthN filter. We can cleanup this filter with new features. We propose how to implement it in this design doc.

Problem

Goals

- As part of Istio moving to unmodified Envoy
- Reimplement Istio authn filter in Wasm if needed
- Current AuthN filter to be slim wasm filter to convert JWT payload to RBAC filter compatible form

Non-Goals

- Revisiting authentication policy

Proposal

Current Status

Current authn filter is implemented a native Envoy filter in istio/proxy. It does following thing:

1. Execute PEER authentication
   - Optional
   - STRICT/PERMISSIVE mode
      - STRICT: if the connection is not TLS, peer has no client certificate, and there is no user, PEER authenticator will reject the request.

- PERMISSIVE: if the connection is not TLS, peer has no client certificate, and there is no user, PEER authenticator will allow the request.

2. Execute ORIGIN authentication
   - Optional
   - Origin authenticaton is built on the top of Envoy JWT AuthN filter. Istio JWT AuthN filter had been required anymore so that it was removed from istio/proxy.
   - The main roles of this phase are
     - to validate extracted JWT's issuer is authorized or not.
     - Build origin authentication result. And pass next filter.
   - It has path based trigger rules.
   - Exchanged token validation.

- What is exchanged token validation?
   - If we configure the extraction location of token as "ingress-authorization", proxy must fetch original claims from jwt payload. Original claims are keyed as "original_claims".

Feature Gap

1. PEER Authentication
   - We need simple spiffe(trust domain validation) layer.
2. ORIGIN Authentication
   - There is no feature to exchange JWT payload for RCToken.
3. There is no simple JWT payload conversion layer for authz(rbac) filter.

Implementation Plan

1. PEER Authentication
   We have two approaches to implement PEER authentication.
   1) Some of functionality will be done with Envoy RBAC filter. We should have API like this on Istio side,

```
{
  “mode”: strict | permissive
}
```

In general, the target to use this condicutation is only mesh-level or namespace level. Istiod will emit such RBAC definition.

STRICT mode

```
{
   "action": "ALLOW",
   "policies": [
      {
         "principal": {
            {
               "any": true
            }
         },
         "permission": {
            {
               "any": true
            }
         },
         "condition": {
            "call_expr": {
               "function": "_&&_",
               "args": {
                  "call_expr": {
                     "function": "_==_",
                     "args": [
                        {
                           "select_expr": {
                              "operand": {
                                 "name": "connection"
                              },
                              "field": "uri_san_local_trust_domain"
                           }
                        },
                        {
                           "select_expr": {
                              "operand": {
                                 "name": "connection"
                              },
                              "field": "uri_san_peer_trust_domain"
                           }
                        }
                     ]
                  },
                  "select_expr": {
                     "operand": {
                        "name": "connection"
                     },
                     "field": "peer_user_name"
                  }
               }
            }
         }
      }
```

```
  ]
}
```

PERMISSIVE mode

```
{
  "action": "ALLOW",
  "policies": [
    {
      "principal": {
        {
          "any": true
        }
      },
      "permission": {
        {
          "any": true
        }
      },
      "condition": {
        "call_expr": {
          "function": "_||_",
          "args": {
            "call_expr": {
              "function": "_==_",
              "args": [
                {
                  "select_expr": {
                    "operand": {
                      "name": "connection"
                    },
                    "field": "uri_san_local_trust_domain"
                  }
                },
                {
                  "select_expr": {
                    "operand": {
                      "name": "connection"
                    },
                    "field": "uri_san_peer_trust_domain"
                  }
                }
              ]
            },
            "select_expr": {
              "operand": {
                "name": "connection"
              },
```

```
                    "field": "peer_user_name"
                }
            }
        }
    }
  ]
}
```

But there is no feature to validate SAN/Local is trust domain. To overcome this problem, we should implement validation layer on CEL runtime and have expose knobs, like uri_san_(peer|local)_trust_domain.

2) Implement SPIFFE filter as wasm extension on istio-proxy. It indicates that we add simple trust domain validation layer on here.

2. ORIGIN Authentication
- Basically we can use jwt_authn filter. But some of feature needs to be implemented.

1) Encapsulated token exchange feature is required on jwt_authn filter
On istio-proxy, there is token exchange functionality to support RCToken validation. For example,

```
{
  "iss": "service@example.com",
  "sub": "user-service",
  "aud": ["user1"],
  "original_claims": {
    "iss": "service2@example.com",
   "sub": "user-service2",
   "aud": ["user1"],
 }
}
```

If we had such JWT payload, envoy should pass encapsulated claims as original payload, like this.

```
{
  "iss": "service2@example.com",
  "sub": "user-service2",
  "aud": ["user1"],
}
```

To enable this feature, we should add this filter config API on jwt_authn,

```
{
  “trigger_header”: [“header1”, “header2”],
  “exchange_claim”: “claim1”
}
```

trigger_header is to specify the target HTTP header to execute token exchange. If header which has specified header on trigger_header arrived, jwt_authn filter will execute token exchange and set claims in a claim specified on exchange_claim.

3. Implement slim conversion layer from raw JWT payload to istio authz compatible form
Istio authz(envoy rbac filter) can’t accept raw JWT payload. So we need to implement simple form conversion layer as wasm extension.

What does control plane send to proxy?

- ORIGIN authenticator
  - istiod sends Envoy jwt_authn filter config to proxy. These configuration is extracted from `security/v1beta1/RequestAuthentication`, which is set by user.
- PEER authenticator
  - istiod sends Envoy mTLS filter configurations. These are extracted from `security/v1beta1/PeerAuthentication`, which is set by user.

Migration path

1. implement to expose TRUST domain and peer user name to CEL runtime in upstream Envoy, or develop simple spiffe id validation filter
2. Add token exchange functionality on JWT AuthN filter
3. Change pilot to send new configuration from istio/api
4. Create tiny wasm filter to construct RBAC filter compatible form following current JwtPayload message.

Appendix