HITACHI
Inspire the Next

CloudNativeSecurityCon NA 2024

How Does a Workload Authenticate an API Request?: Implementing Transaction Tokens with Keycloak

06/27/2024

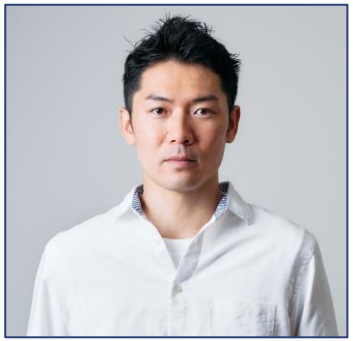
Yoshiyuki Tabata

OSS Solution Center
Hitachi, Ltd.

© Hitachi, Ltd. 2024. All rights reserved.

About the speaker

HITACHI
Inspire the Next



Yoshiyuki Tabata

- Senior OSS Consultant at Hitachi, Ltd.
- CNCF Ambassador / Cloud Native Community Japan organizer / Cloud Native Security Japan founder
- LinkedIn: @ytabata, X: @yo_tabata, GitHub: @y-tabata

- Specialist in API authorization
  - Consulting for API management infrastructure and authentication/authorization systems in the financial, public, social, and industrial fields
- Contributor to OSS related to authentication, authorization, and API management
  - Keycloak (IAM OSS)
  - 3scale (API management OSS)
- Other activities
  - Speaker at events such as Apidays, API Specifications Conference, OAuth Security Workshop, etc.
  - Author of Keycloak books (Japanese) and writer of web articles about IAM (Japanese)

© Hitachi, Ltd. 2024. All rights reserved.

HITACHI
Inspire the Next

Contents

1. Difficulty for workloads to authenticate API requests
2. Introduction of Transaction Tokens
3. Implementing Transaction Tokens with Keycloak
4. Demonstration

© Hitachi, Ltd. 2024. All rights reserved.

HITACHI
Inspire the Next

Contents

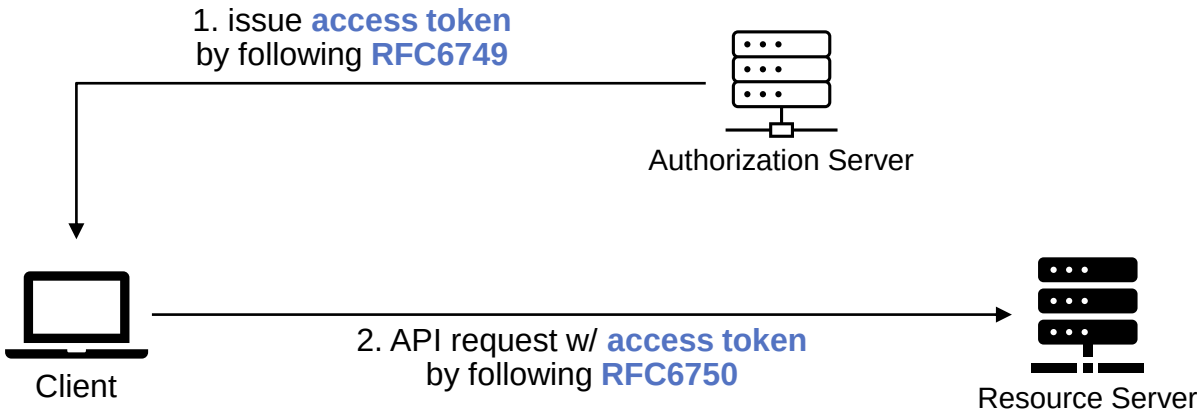
1. Difficulty for workloads to authenticate API requests
2. Introduction of Transaction Tokens
3. Implementing Transaction Tokens with Keycloak
4. Demonstration

© Hitachi, Ltd. 2024. All rights reserved.

De facto standard for API authentication: OAuth 2.0 (RFC6750)

HITACHI
Inspire the Next

- **RFC6750** (The OAuth 2.0 Authorization Framework: Bearer Token Usage) describes "the client requests the protected resource from the resource server and authenticates by presenting the **access token**," and is the most common way to authenticate API requests.
- The way of issuing an access token is described in **RFC6749** (The OAuth 2.0 Authorization Framework).

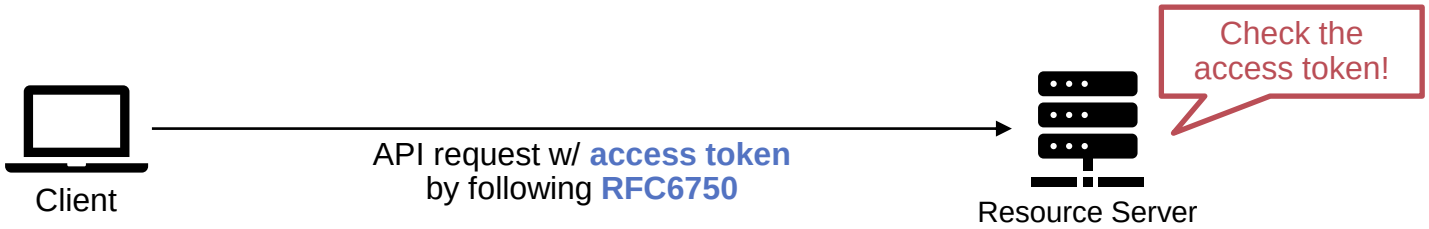


© Hitachi, Ltd. 2024. All rights reserved.

Recommended checks of OAuth access token

HITACHI
Inspire the Next

- To authenticate an API request, the resource server should perform at least the following checks of the OAuth access token. These are recommended by such as **RFC6750**, and also by the draft **CNCF Zero Trust Whitepaper**[*1].
  - Signature (if assertion-based tokens like JWT): Verify if the token has not been tampered with.
  - exp and nbf claims: Validate the expiration time of the token.
  - iss claim: Confirm the issuer's identity.
  - aud claim: Check if the audience includes the resource server.
  - scope claim (if exists): Ensure that the resource owner authorizes access to the resource.

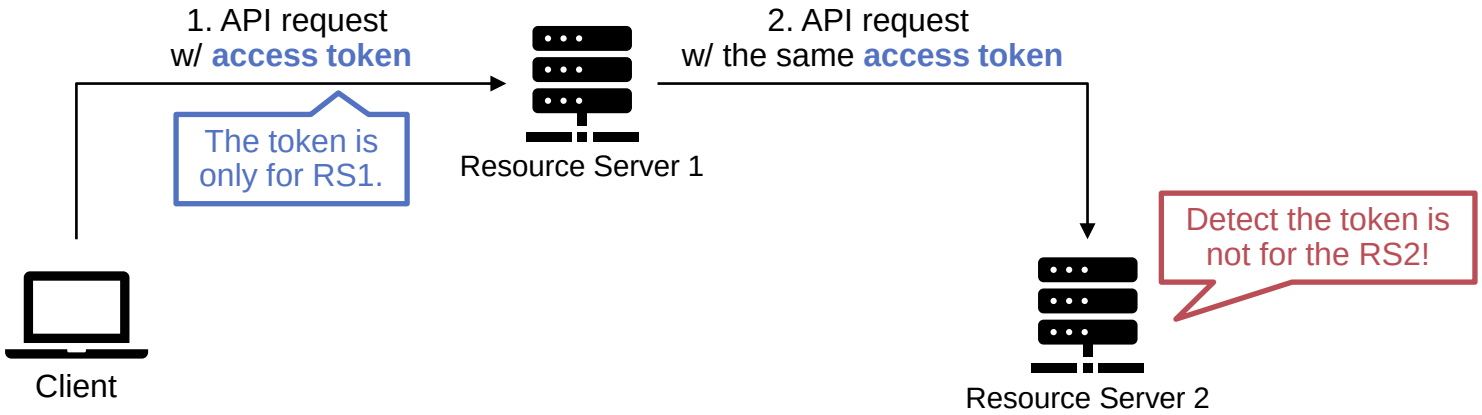


*1: CNCF Zero Trust Whitepaper: https://github.com/cncf/tag-security/pull/1229

© Hitachi, Ltd. 2024. All rights reserved.

Token Redirect attack

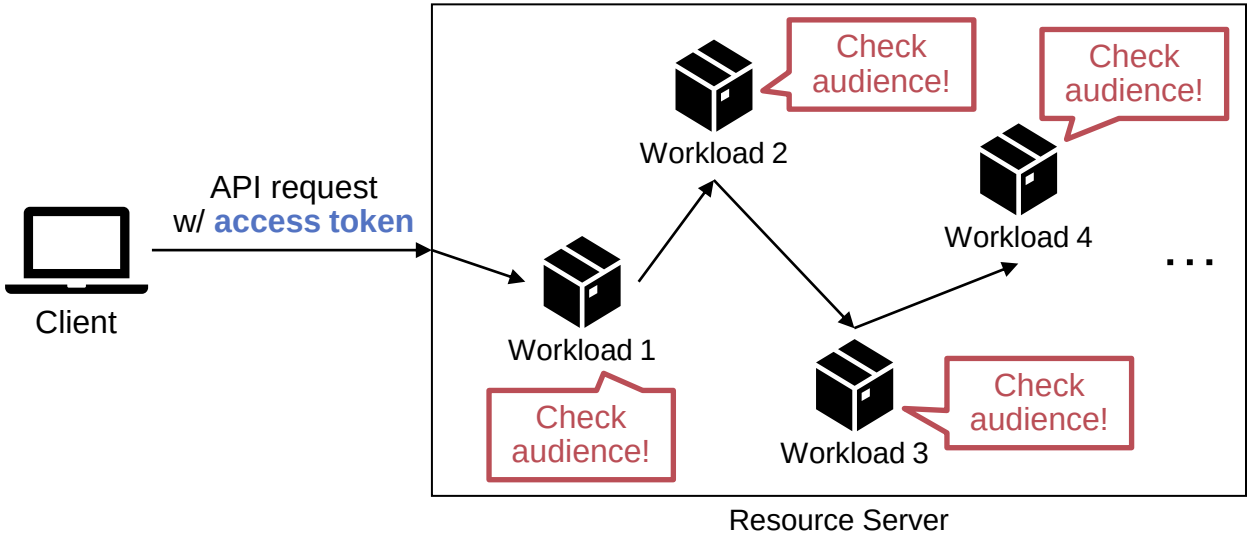
- **Token redirect attack** is an attack in which a token generated for consumption by one resource server is used to gain access to a different resource server.
- Among the token checks, the **audience** check is to deal with **token redirect attacks**.
  - aud claim: Check if the audience includes the resource server.



© Hitachi, Ltd. 2024. All rights reserved.

Difficulty for workloads to check "Audience"

- In cloud-native architectures, there are **many internal workloads**, and each workload should check the **audience** is the workload itself or not, especially in **zero trust context**.
- The **audience** is decided based on **the resource owner's consent**, but it's hard for the resource owner to identify all audiences and consent for each consumption.



© Hitachi, Ltd. 2024. All rights reserved.

HITACHI
Inspire the Next

Contents

1. Difficulty for workloads to authenticate API requests
2. Introduction of Transaction Tokens
3. Implementing Transaction Tokens with Keycloak
4. Demonstration

© Hitachi, Ltd. 2024. All rights reserved.

Transaction Tokens (Txn-Tokens)

HITACHI
Inspire the Next

- Txn-Tokens are discussed in **IETF OAUTH WG**, and its internet draft is published draft-ietf-oauth-transaction-tokens-01.
- Txn-Tokens are **short-lived, signed JWTs** that assert the identity of a user or a workload and assert an authorization context.

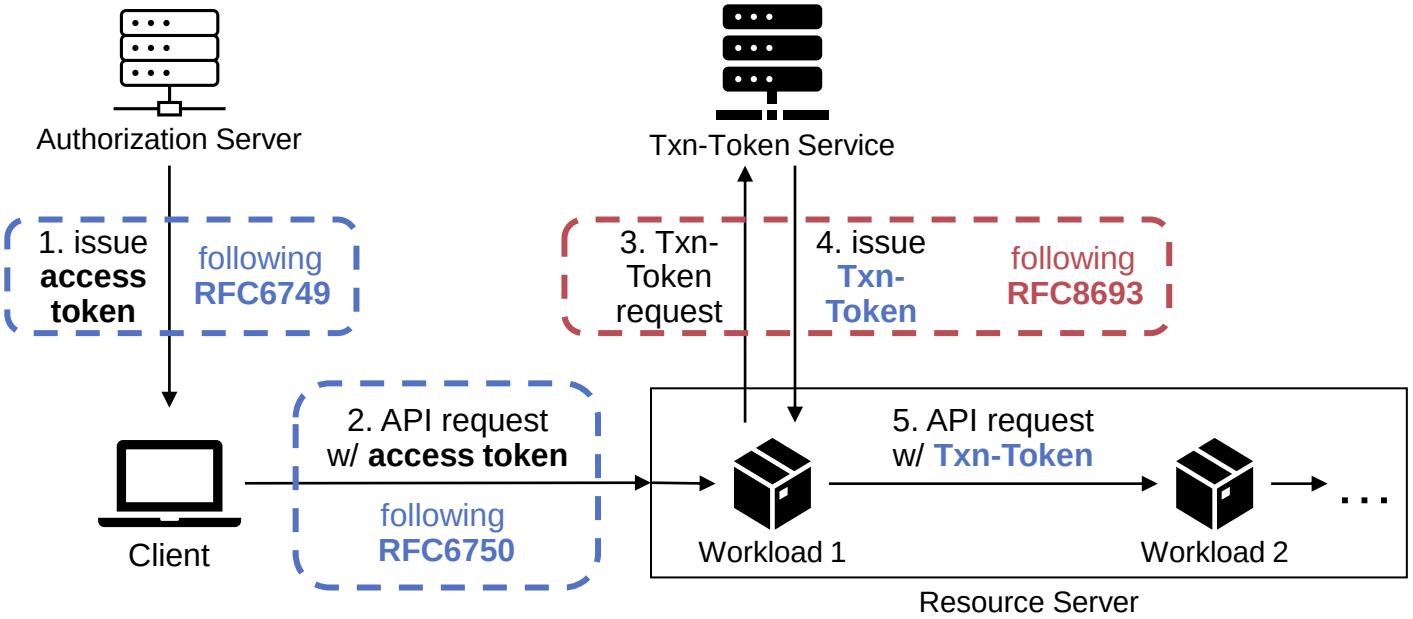
Txn-Token body example

```
{
   "iat": "1686536226000",
   "aud": "trust-domain.example", // audience
   "exp": "1686536526000",
   "txn": "97053963-771d-49cc-a4e3-20aad399c312",
   "sub": "d084sdrt234fsaw34tr23t",
   "rctx": {
     "req_ip": "69.151.72.123", // env context of external call
     "authn": "urn:ietf:rfc:6749", // env context of the external call
     "req_wl": "apigateway.trust-domain.example" // the internal entity that requested the Txn-Token
   },
   "purp" : "trade.stocks", // purpose or intent of this transaction
   "azd": { … } // authorization context details
}
```

© Hitachi, Ltd. 2024. All rights reserved.

Transaction Tokens (Txn-Tokens)

- A workload performs an **OAuth 2.0 Token Exchange (RFC8693)** to obtain a Txn-Token, by invoking a special Token Service (the **Txn-Token Service**) and provides sufficient context to generate a Txn-Token.

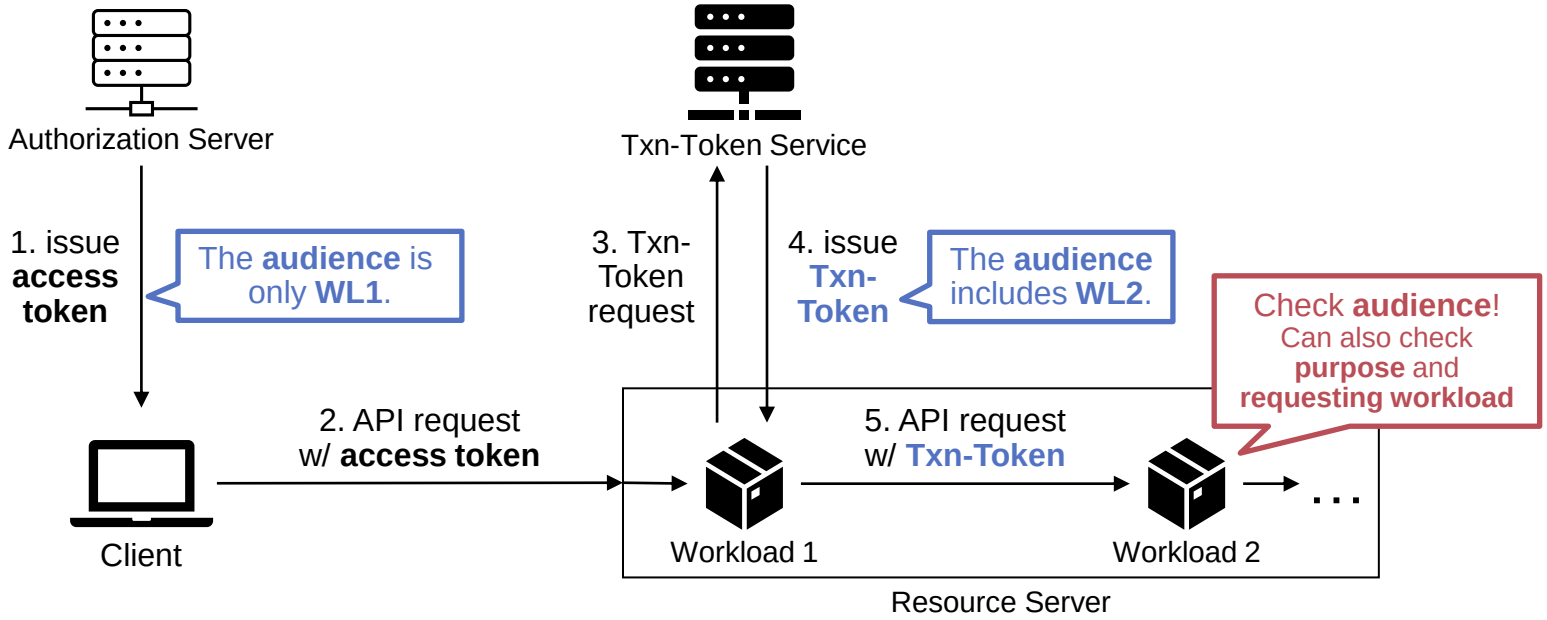


© Hitachi, Ltd. 2024. All rights reserved.

Transaction Tokens (Txn-Tokens)

HITACHI
Inspire the Next

- Txn-Tokens help **prevent spurious invocations** by ensuring that a workload receiving an invocation can independently verify the user or workload on whose behalf an external call was made and any context relevant to the processing of the call.



© Hitachi, Ltd. 2024. All rights reserved.

HITACHI
Inspire the Next

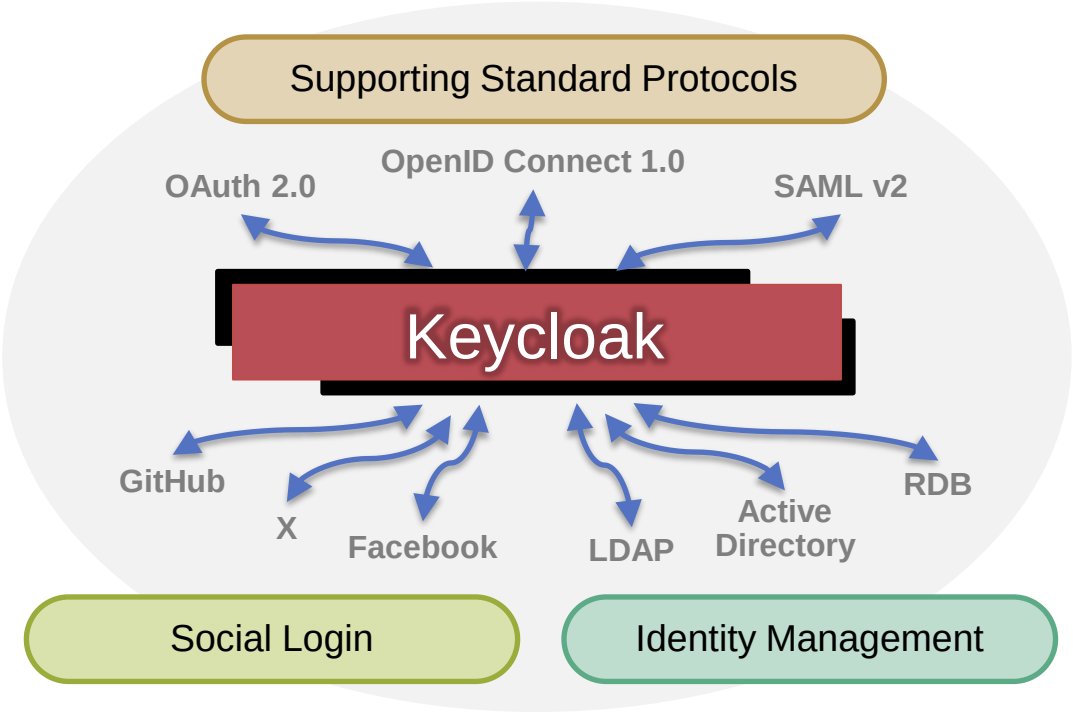
Contents

1. Difficulty for workloads to authenticate API requests
2. Introduction of Transaction Tokens
3. Implementing Transaction Tokens with Keycloak
4. Demonstration

© Hitachi, Ltd. 2024. All rights reserved.

Keycloak

- Keycloak is IAM (Identity and Access Management) OSS.
- Keycloak provides OAuth2 authorization server features and single sign-on features.
- Keycloak is a CNCF incubating project.



Major features

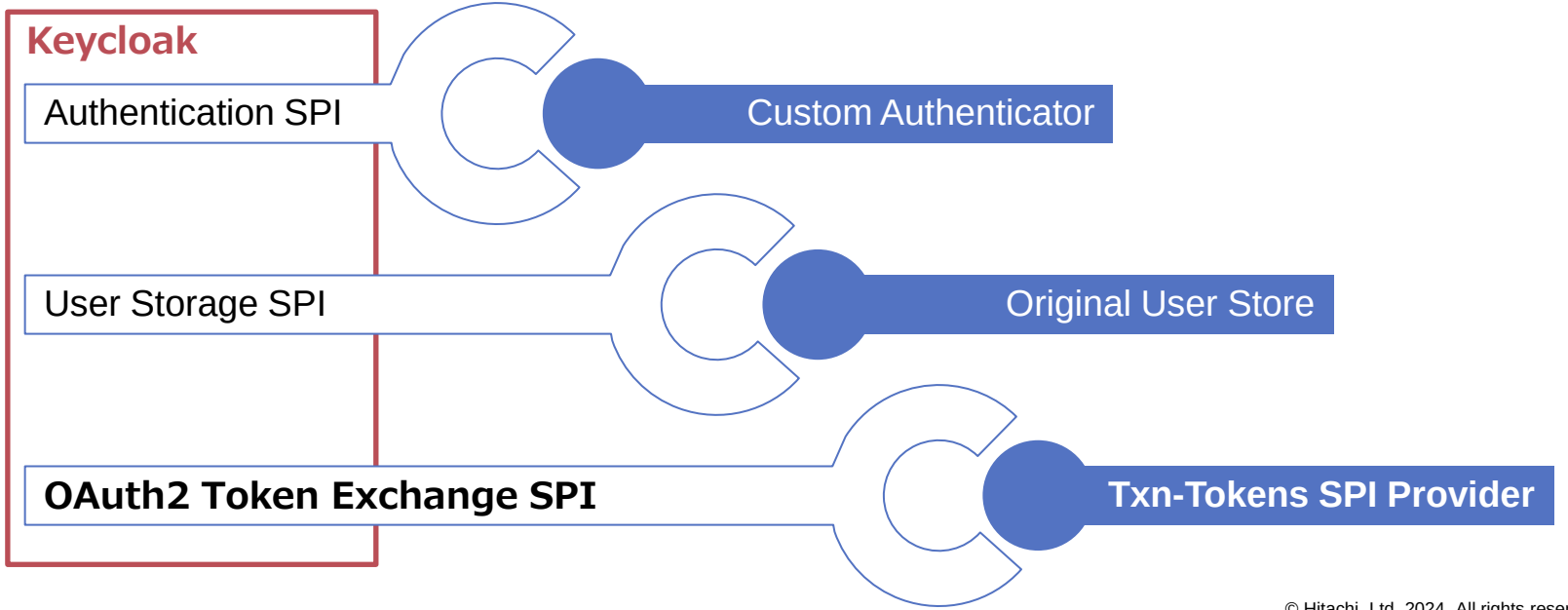
- Supporting standards. ex. OAuth 2.0, OpenID Connect 1.0, SAML v2, **OAuth 2.0 Token Exchange (Preview)**, …
- Login with social networks.
- Connect to existing user stores. ex. LDAP, Active Directory, …

© Hitachi, Ltd. 2024. All rights reserved.

Service Provider Interfaces (SPI)

HITACHI
Inspire the Next

- Keycloak is designed to cover most use cases without requiring custom code but is also flexibly customizable.
- To achieve this Keycloak has many **Service Provider Interfaces (SPI)** for which you can introduce your custom features without building Keycloak source code.
- In Keycloak, there are **over 100 SPIs**, and there are **over 400 built-in SPI providers**.

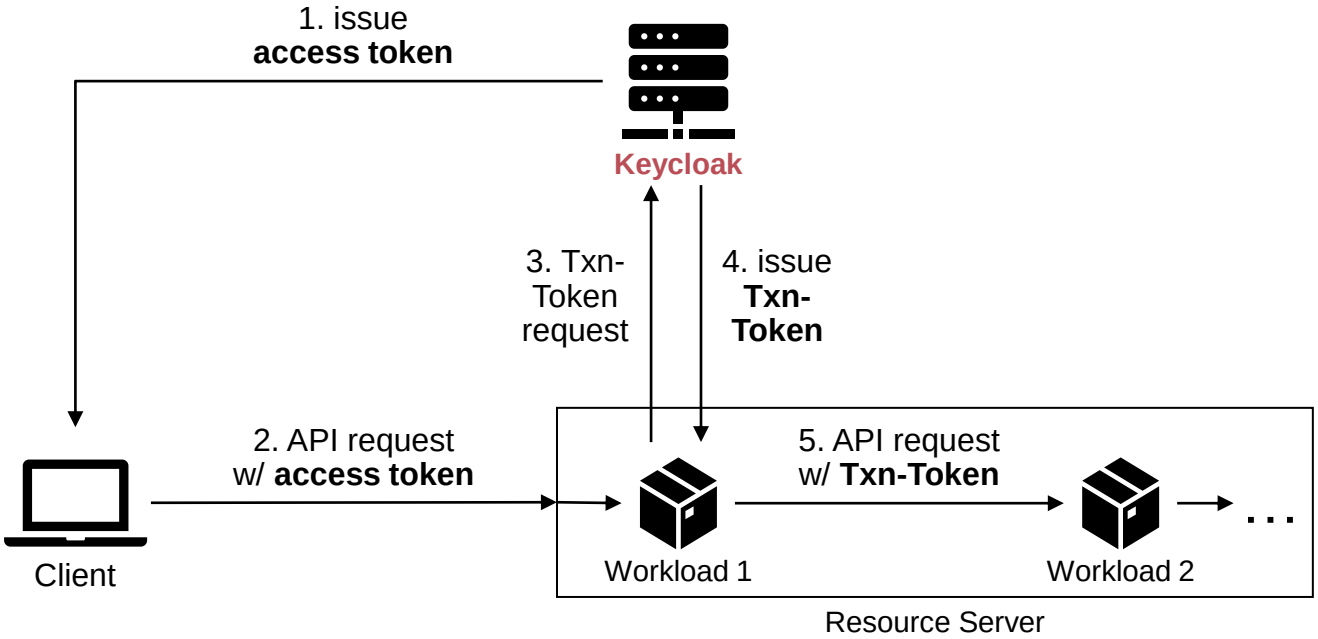


© Hitachi, Ltd. 2024. All rights reserved.

Implementing Transaction Tokens with Keycloak

HITACHI
Inspire the Next

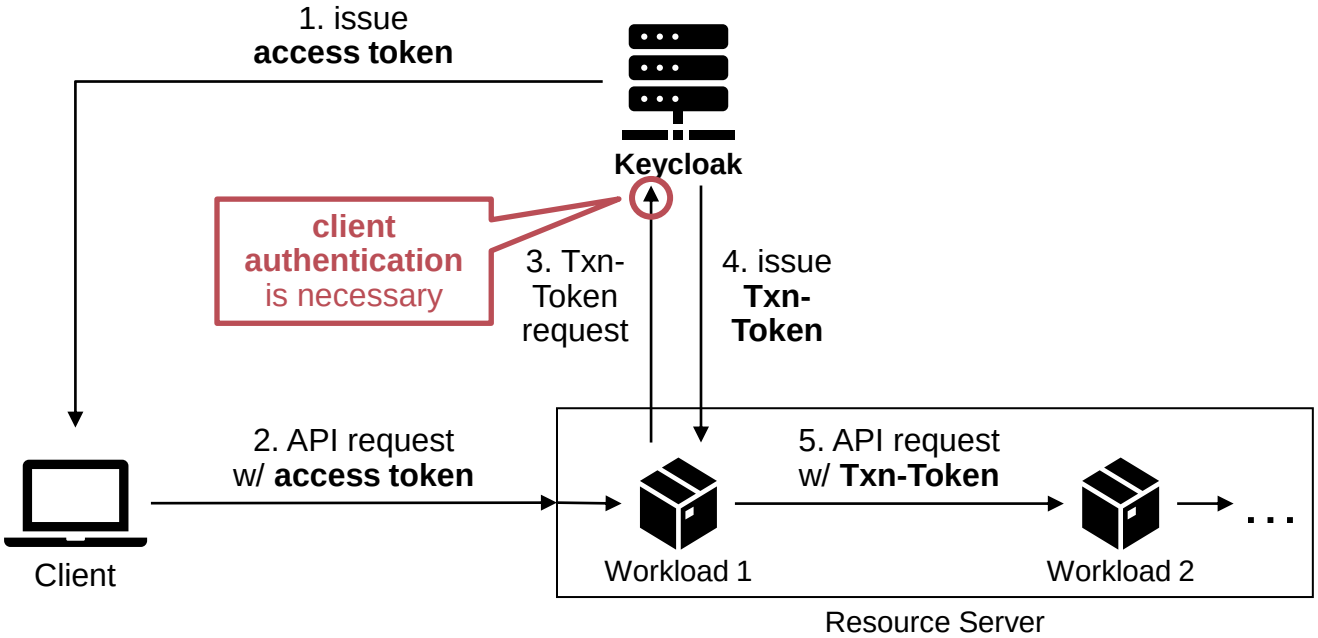
- By implementing an SPI provider and **extending token exchange feature**, Keycloak can have Txn-Token service features.
- One **Keycloak** can play two roles: **OAuth2 authorization server** and **Txn-Token service**.



© Hitachi, Ltd. 2024. All rights reserved.

Implementer's consideration: client authentication

- As the same OAuth 2.0 Token Exchange (RFC8693), **client authentication** is necessary to issue Txn-Token.
- The way of client authentication is **out of the scope** of the specification but recommended to rely on mechanisms such as **SPIFFE**.



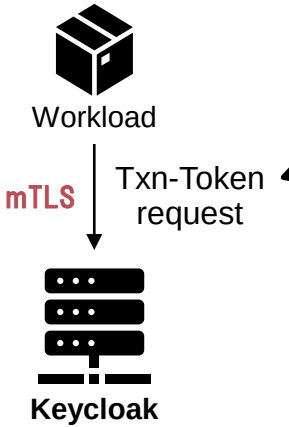
© Hitachi, Ltd. 2024. All rights reserved.

Implementer's consideration: client authentication

HITACHI
Inspire the Next

- There are 2 formats for **SPIFFE Verifiable Identity Document (SVID)**: an X.509 certificate (**X.509-SVID**), and a JWT token (**JWT-SVID**).

- **X.509-SVID**
  - Client authentication occurs with mutual TLS utilizing the PKI method of associating a certificate to a client, similar to **tls_client_auth** defined by **RFC8705**.




```
POST /txn-token-service/token_endpoint HTTP 1.1
Host: txn-token-service.trust-domain.example
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Atoken-exchange
&requested_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-type%3Atxn-token
&audience=http%3A%2F%2Ftrust-domain.example
&scope=finance.watchlist.add
&subject_token=<<access token>>
&subject_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-type%3Aaccess_token
&request_context=eyAiaXBfYWRkcmVzcyI6ICIxMjcuMC4wLjEiLCAiY2xpZW50IjogIm1vY
mlsZS1hcHAiLCAiY2xpZW50X3ZlcnNpb24iOiAidjExIiB9
&client_id=<<client ID>>
```

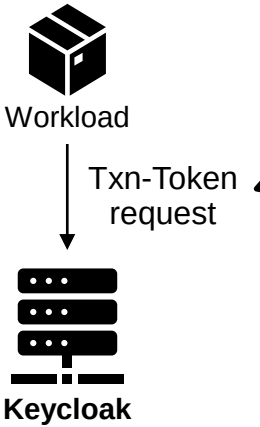
© Hitachi, Ltd. 2024. All rights reserved.

Implementer's consideration: client authentication

HITACHI
Inspire the Next

- There are 2 formats for **SPIFFE Verifiable Identity Document (SVID)**: an X.509 certificate (**X.509-SVID**), and a JWT token (**JWT-SVID**).

- **JWT-SVID**
  - Client authentication occurs with a JWT bearer token. In the context of **OAuth 2.0 token exchange**, using the **actor_token** parameter.




```
POST /txn-token-service/token_endpoint HTTP 1.1
Host: txn-token-service.trust-domain.example
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Atoken-exchange
&requested_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-type%3Atxn-token
&audience=http%3A%2F%2Ftrust-domain.example
&scope=finance.watchlist.add
&subject_token=<<access token>>
&subject_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-type%3Aaccess_token
&request_context=eyAiaXBfYWRkcmVzcyI6ICIxMjcuMC4wLjEiLCAiY2xpZW50IjogIm1vY
mlsZS1hcHAiLCAiY2xpZW50X3ZlcnNpb24iOiAidjExIiB9
&actor_token=<<JWT-SVID>>
&actor_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-type%3Ajwt
```

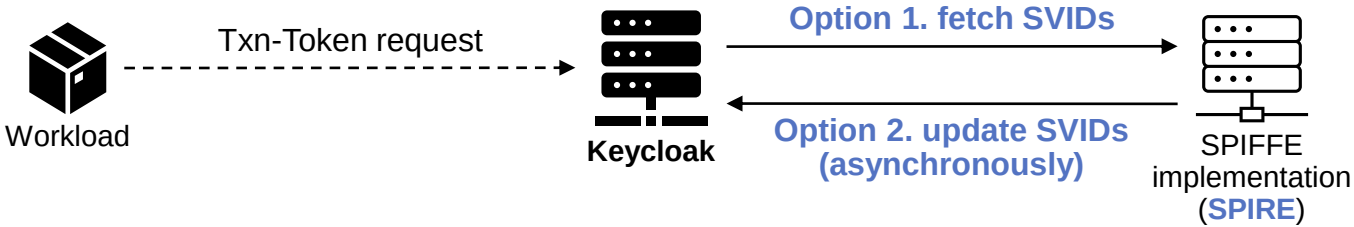
© Hitachi, Ltd. 2024. All rights reserved.

Implementer's consideration: client authentication

- How to verify SVIDs is another consideration. To verify SVIDs, Keycloak needs the latest SVIDs which are **rotated frequently and automatically**.

- **Option 1**: Keycloak **fetches SVIDs from SPIFFE Workload Endpoint** every time Keycloak receives a Txn-Token request.

- **Option 2**: **Trigger a SVID rotation** and update the credential Keycloak has.



© Hitachi, Ltd. 2024. All rights reserved.

HITACHI
Inspire the Next

Contents

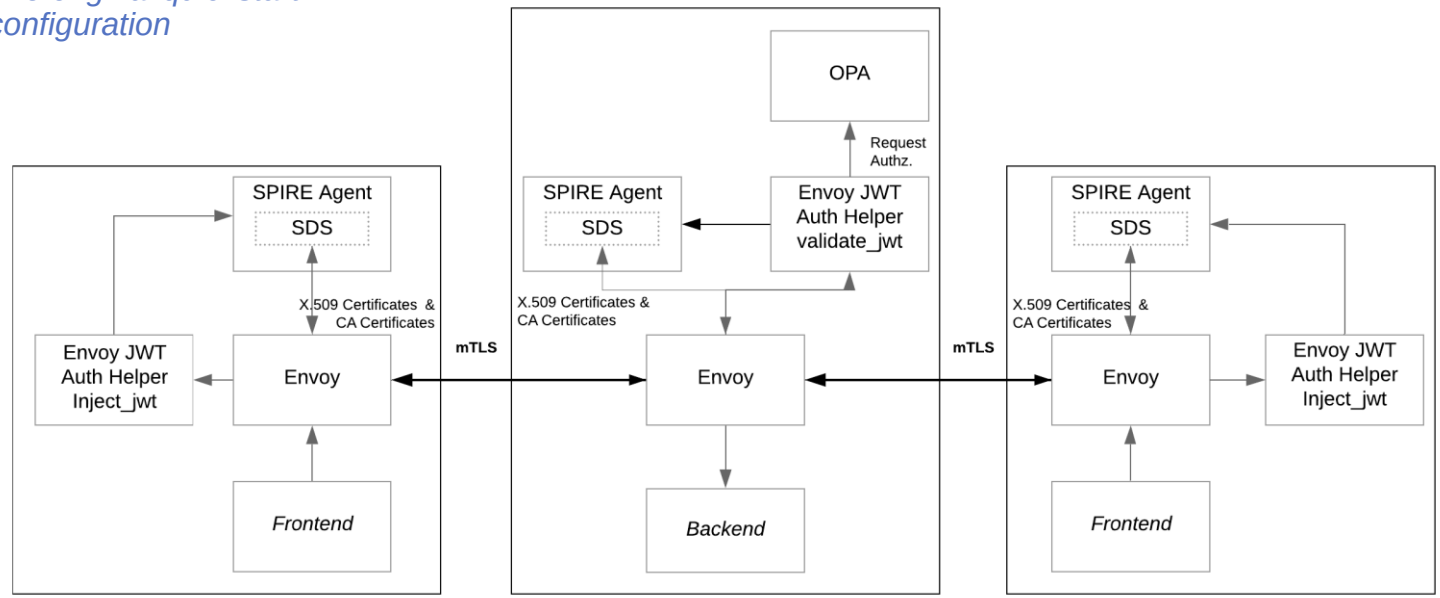
1. Difficulty for workloads to authenticate API requests
2. Introduction of Transaction Tokens
3. Implementing Transaction Tokens with Keycloak
4. Demonstration

© Hitachi, Ltd. 2024. All rights reserved.

Demonstration configuration

- The configuration is based on **SPIFFE quickstart**.
- Here, we use **JWT-SVID**, because its implementation is easier than X.509-SVID.

The original quickstart configuration

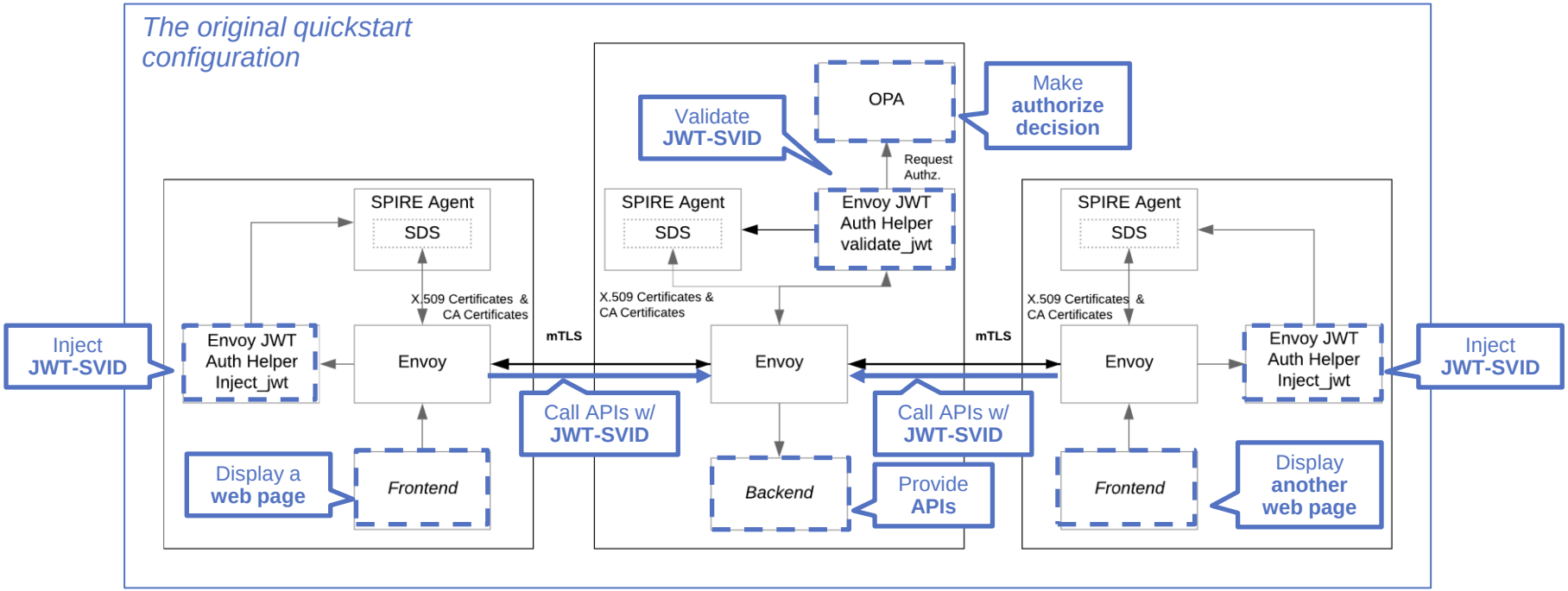


OPA Authorization with Envoy and JWT-SVIDs: https://spiffe.io/docs/latest/microservices/envoy-jwt-opa/readme/

© Hitachi, Ltd. 2024. All rights reserved.

Demonstration configuration

- The configuration is based on **SPIFFE quickstart**.
- Here, we use **JWT-SVID**, because its implementation is easier than X.509-SVID.

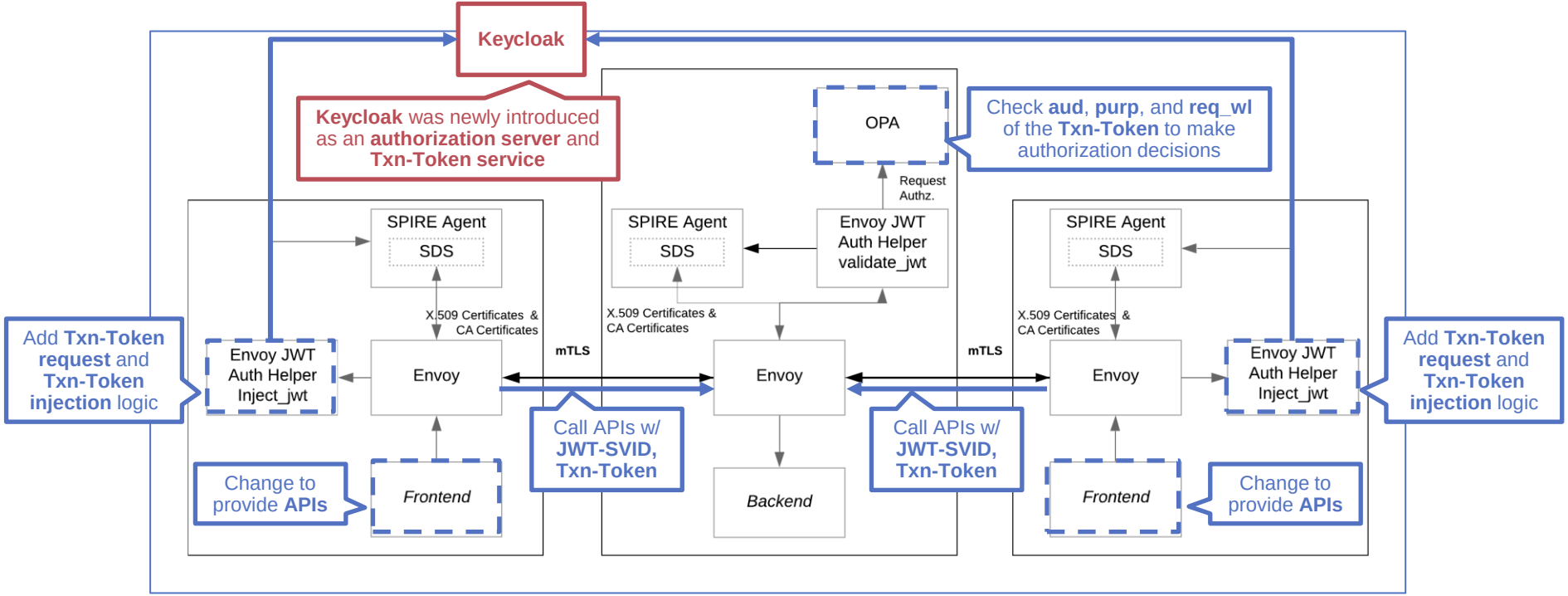


OPA Authorization with Envoy and JWT-SVIDs: https://spiffe.io/docs/latest/microservices/envoy-jwt-opa/readme/

© Hitachi, Ltd. 2024. All rights reserved.

Demonstration configuration

- The configuration is based on **SPIFFE quickstart**.
- Here, we use **JWT-SVID**, because its implementation is easier than X.509-SVID.

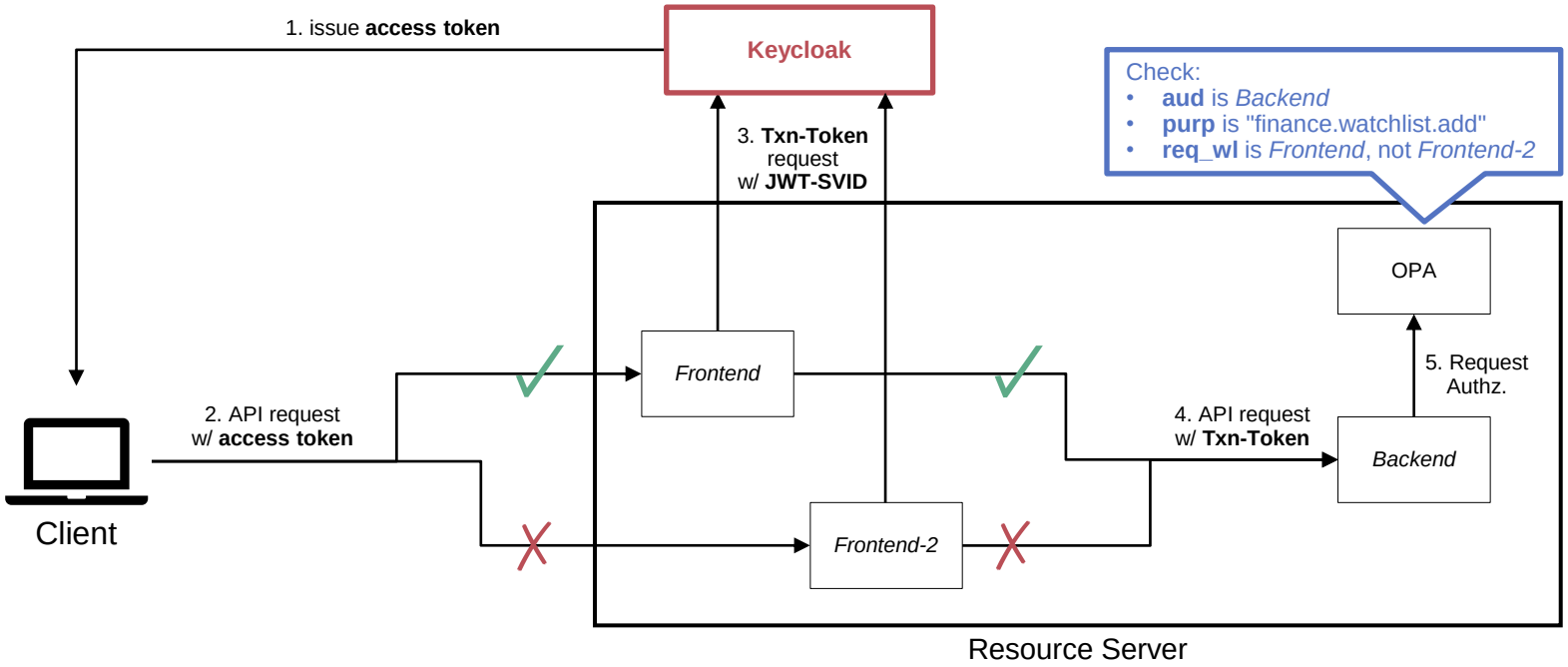


OPA Authorization with Envoy and JWT-SVIDs: https://spiffe.io/docs/latest/microservices/envoy-jwt-opa/readme/

© Hitachi, Ltd. 2024. All rights reserved.

Demonstration configuration

- The configuration is based on **SPIFFE quickstart**.
- Here, we use **JWT-SVID**, because its implementation is easier than X.509-SVID.



© Hitachi, Ltd. 2024. All rights reserved.

Summary

HITACHI
Inspire the Next

- We implemented **Txn-Tokens** in **Keycloak** and demonstrated it.
- There is room for consideration of **client authentication**.
  - There are 2 formats for SVID: **X.509-SVID** and **JWT-SVID**.
    - In the demo, we used **JWT-SVID**.
      - If using X.509-SVID, we must consider how to receive X.509-SVID in Keycloak.
      - Adding the **Txn-Token** and the **JWT-SVID** to the API request is a bit redundant, so it might be better to consider integrating the Txn-Token service with the SVID provider.

  - How to verify SVIDs is another consideration. To verify SVIDs, Keycloak needs the latest SVIDs which are rotated frequently and automatically.
    - There are 2 options: **fetching SVIDs from the SPIFFE Workload Endpoint** or **triggering an SVID rotation**.
      - These are future tasks.

© Hitachi, Ltd. 2024. All rights reserved.

Trademarks

- OpenID is a trademark or registered trademark of OpenID Foundation in the United States and other countries.
- GitHub is a trademark or registered trademark of GitHub, Inc. in the United States and other countries.
- Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries.
- X is a trademark or registered trademark of X Corp. in the United States and other countries.
- Facebook is a trademark or registered trademark of Meta Platforms, Inc. in the United States and other countries.
- Other brand names and product names used in this material are trademarks, registered trademarks, or trade names of their respective holders.

© Hitachi, Ltd. 2024. All rights reserved.

HITACHI
Inspire the Next

END

How Does a Workload Authenticate an API Request?: Implementing Transaction Tokens with Keycloak

06/27/2024

Yoshiyuki Tabata

OSS Solution Center

Hitachi, Ltd.

© Hitachi, Ltd. 2024. All rights reserved.


Hitachi Social Innovation is
POWERING GOOD