TOP-KT-021 - Token Introspection

Versiegeschiedenis...

Versie	Datum	Status	Wijzigingen
1.0.1	26 Jun 2023	definitief	Dead link aangepast
1.0.0	24 Apr 2023	definitief	

Beschrijving

Het token introspection endpoint is een onderdeel van de autorisatieservice dat op basis van OAuth 2.0 Token Introspection (rfc7662) aan de applicaties in de infrastructuur de mogelijkheid biedt JWT tokens te valideren. Deze JWT tokens kunnen zijn:

- HTI launch tokens
- Het `access_token` van de authorisatieservice.
- Het `id_token` van de authorisatieservice.

De autorisatieservice heeft toegang tot de domeinconfiguratie en met de informatie die daar aanwezig is de JWT tokens correct valideren.

Overwegingen

RFC 7662, 200 OK

De OAuth 2.0 Token Introspection (rfc7662) werkt met een HTTP endpoint waar een JWT token ter validatie naartoe gestuurd kan worden. De response bestaat uit:

- Een response met een JSON map met daarin de active: false waarde indien het token niet valide is.
- Een response met een JSON map met daarin de active: true waarde samen met de inhoud van de body van het JWT token indien het token valide is.

De response code van beide antwoorden zijn 200 OK, wat een punt van aandacht is. Het introspection endpoint kan een 4xx of 5xx error teruggeven op het moment er iets anders misgaat met het request, maar deze heeft niets te maken met de inhoud van de JWT token die gevalideerd wordt.

Publicatie van het token introspection endpoint

Het token introspection endpoint wordt gepubliceerd in de smart configuratie op bij de FHIR service (.well-known/smart-configuration). Deze configuratie bevat meerdere elementen, waarvan voor de token introspectie de waarde van het `introspection_endpoint` van belang is.

Gebruik van het token introspection endpoint

Applicaties in een koppeltaaldomein kennen elkaar niet, en zijn niet op de hoogte van de JWKS endpoints van elkaar in het domein. Dit is een bewuste keuze van Koppeltaal, het beheer van het domein wordt door domeinbeheer uitgevoerd, deze geeft aan wie er deelneemt en met welke beperkingen en voorwaarden. Het is dus domeinbeheer die de JWKS endpoints bijhoudt en het token introspection endpoint die gebruikt dient te worden om de tokens te valideren. Het is dus voor individuele applicaties in het domein niet mogelijk zelf een HTI token, `access_token` of `id_token` te valideren.

Beveiliging van het token introspection endpoint

Het token introspection endpoint moet volgens de specificatie beveiligd worden om te beveiligen tegen token scanning aanvallen. In koppeltaal maken we de keuze de beveiliging te doen op basis van JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication (rfc7523) zoals bij de SMART on FHIR backend services beschreven TOP-KT-005c - Applicatie toegang: SMART on FHIR backend services. Hiermee kiezen we voor een uniforme aanpak rond client authenticatie. Om het token introspection endpoint aan te kunnen spreken moet het aanroepende systeem een zelf ondertekende JWT meesturen als client identificatie.

Droste effect

Het gevolg van rfc7523 is dat er een eigen JWT token nodig is om een JWT token te valideren. Dit lijkt omslachtig en geeft het gevoel van een droste effect.

Echter, er zit logica achter:

1. **Beveiliging**: het token introspection endpoint MOET beveiligd worden, anders wordt het mogelijk een doelwit voor verschillende aanvallen.

2. **Consistentie**: de rfc7523 wordt verder voor alle client identificatie / authenticatie gebruikt, zowel for SMART on FHIR backend services als voor SMART on FHIR app launch.
3. **Hergebruik**: in het verlengde van het vorige punt, eenmaal geïmplementeerd is het (her-) bruikbaar voor all communicatie met de autorisatieservice.

Toepassing, restricties en eisen

Token introspection

JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication

Het token introspection endpoint wordt beveiligd door middel van JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication (rfc7523) zoals beschreven in TOP-KT-005c - Applicatie toegang: SMART on FHIR backend services. In het kort zien de stappen er als volgt uit.

De client genereert een JWT met de volgende waarden en ondertekent deze met het eigen private sleutel:

- `iss` (Issuer): de client_id van de applicatie
- `sub` (Subject): de client_id van de applicatie
- `aud` (Audience): de URL van het introspection endpoint
- `iat` (Issued At): timestamp van uitgifte
- `exp` (Expiration): geldig tot timestamp, maximaal 5 minuten
- `jti` (JWT ID): unieke waarde per token

Vervolgens wordt er aan het request de volgende parameters toegevoegd:

- `client_assertion_type` : `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`
- `client_assertion` : het gegenereerde JWT token uit de vorige stap.

Het uitvoeren van de token introspection in stappen

Voordat het token introspection endpoint aangesproken kan worden gaan we er vanuit dat de volgende situatie van toepassing is:

- De applicatie is aangemeld in het domein en heeft zijn public key bekend gemaakt door een JWKS url of door een directe registratie van de public key.
- In de FHIR resource service moet een Device zijn aangemaakt waarin de client_id overeenkomt met de applicatie.

Het aanspreken van het token introspection endpoint bestaat uit de volgende stappen.

De module applicatie haalt het smart configuratie op bij de FHIR service (.well-known/smart-configuration). Dit statement bevat de volgende endpoints:

- `introspection_endpoint`

De applicatie genereert een JWT token voor de client_assertion ondertekend met de private key met de volgende velden:

- `iss` (Issuer): de client_id van de applicatie
- `iat` (Issued At): de timestamp now()
- `jti` (JWT ID): een unieke waarde voor deze aanvraag, UUIDs zijn hiervoor geschikt.
- `exp` (Expiration Time): de timestamp now() + 5 minuten

> **Twee JWT tokens**
>
> Om verwarring te voorkomen, er zijn op dit moment twee JWT tokens: de eigen genereerde JWT token voor de client_assertion en het JWT token die de applicatie wil valideren. Dit laatste kan bijvoorbeeld het HTI launch token zijn, maar ook access_tokens kunnen bij het introspection endpoint gevalideerd worden.

De applicatie benadert het introspection_endpoint endpoint met de volgende parameters in een application/x-www-form-urlencoded POST request:

- `token` : de inhoud van het te valideren JWT token.
- `client_assertion_type` : `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`
- `client_assertion` : het gegenereerde JWT token uit de vorige stap.

Het token introspection endpoint checkt de **client assertion** met de volgende stappen:

- De `client_assertion` JWT wordt uitgepakt zonder validatie.
- In het `iss` (Issuer) wordt de `client_id` van de ondertekende applicatie verwacht.
- De applicatie wordt op basis van de `client_id` in de domeinconfiguratie opgezocht. Deze bevat een URL voor het JWKS endpoint of de waarde van de publieke sleutel.
- Indien er een JWKS endpoint wordt gebruikt, wordt deze bevraagd, de header van de JWT moet een Key ID (kid) bevatten. Een publieke sleutel met de Key ID van het JWT token moet voorkomen in het JWKS bestand.
- De JWT token wordt gevalideerd met de publieke sleutel, tevens wordt de validatie volgens de JWT standaard validatie uitgevoerd. Daaronder valt:
-
    - De check op de `jti` (JWT ID).
    - De check op de verwachte `aud` (Audience), de URL van het token introspection endpoint.
    - Het `exp` (Expiration Time) veld, en het nbf (Not Before) indien aanwezig.

- Indien het token niet valide is, wordt een status 401 Unauthorized teruggegeven.

Het token introspection endpoint voert de **validatie** van het te valideren JWT token uit in de volgende stappen:

- De token JWT token wordt uitgepakt zonder validatie.
- In het `iss` (Issuer) veld wordt de client_id van de ondertekende applicatie verwacht.
- De applicatie wordt op basis van de `client_id` in de domeinconfiguratie opgezocht. Deze bevat een URL voor het JWKS endpoint of de waarde van de publieke sleutel.
- Indien er een JWKS endpoint wordt gebruikt, wordt deze bevraagd, de header van de JWT moet een Key ID (`kid`) bevatten. Een publieke sleutel met de Key ID van het JWT token moet voorkomen in het JWKS bestand.
- De JWT token wordt gevalideerd met de publieke sleutel, tevens wordt de validatie volgens de JWT standaard validatie uitgevoerd. Daaronder valt:
-
  - De check op de `jti` (JWT ID).
  - De check op de verwachte `aud` (Audience).
  - De `exp` (Expiration Time) en `nbf` (Not Before) velden.

Na het succesvol valideren van het token wordt de inhoud van de body teruggestuurd, met als toevoeging de active waarde op true, als het JWT token niet valide is, wordt enkel een JSON map met active waarde als false teruggestuurd, zoals de OAuth 2.0 Token Introspection (rfc7662) voorschrijft.

Gebruikte standaarden

OAuth 2.0 Token Introspection (rfc7662): https://www.rfc-editor.org/rfc/rfc7662.

JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants: https://www.rfc-editor.org/rfc/rfc7523

Voorbeelden

Een voorbeeld van een valide response.

Valide response

```
HTTP/1.1 200 OK
Content-Type: application/json
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzUxMiJ9.eyJpc3MiOiIwODcwOWUzNy0zY2EwLTQ1NjEtODQ0MC1lM2IyOD
E2NGNlOWUiLCJzdWIiOiIwODcwOWUzNy0zY2EwLTQ1NjEtODQ0MC1lM2IyODE2NGNlOWUiLCJhdWQiOiJodHRwO
i8vbG9jYWxob3N0OjgwODAvZW5kcG9pbnQiLCJpYXQiOjE2Njc5MDY0MjEsImV4cCI6MTY2NzkwNjcyMSwianRp
IjoiZWMzM2NjYjctNjk2MC00MmQ1LWJmNjItZGU5NTc5NDQzMjg4In0.bakizOU_hXgIAlqN_U-
UZoOLW06gsphRqxNIsWoR4IeY6Wk-ZmZ7SYPLhVxsOzZxX-
A6NVCj7fbxgmpkZAeWdkYSQj53GtFvLrXyfyu0gjxcOLqk9yg-V7DQkTzro_R9bDEe9p-x5BX2Sky-
```

```
06gZQbPI23OZiwB-
7BL4F0Y15cwgmgA1mE1e_i96okg2Y3BkzNbOTkD1jpkgKRjws6bW77AyOL96e6LyM2b_Yo5XyAcp1Za-PG-
r2cEjS7Xbz0N4uvRtHuvFt5uB8fa4YnrF_5e2eBRPfrgnbqpjXwLHD7MA1nWd2FEyKRqhHafkj5Z6yeyhXVpYQe
Ss2Db63efRGA

{
 "active": true,
 "client_id": "l238j323ds-23ij4",
 "username": "jdoe",
 "scope": "read write dolphin",
 "sub": "Z5O3upPC88QrAjx00dis",
 "aud": "https://protected.example.net/resource",
 "iss": "https://server.example.com/",
 "exp": 1419356238,
 "iat": 1419350238,
 "extension_field": "twenty-seven"
}
```

Een voorbeeld van een invalide response.

Invalide response

```
HTTP/1.1 200 OK
Content-Type: application/json
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzUxMiJ9.eyJpc3MiOiIwODcwOWUzNy0zY2EwLTQ1NjEtODQ0MC11M2IyODE
2NGNlOWUiLCJzdWIiOiIwODcwOWUzNy0zY2EwLTQ1NjEtODQ0MC11M2IyODE2NGNlOWUiLCJhdWQiOiJodHRwOi8
vbG9jYWxob3N0OjgwODAvZW5kcG9pbnQiLCJpYXQiOjE2Njc5MDY0MjEsImV4cCI6MTY2NzkwNjcyMSwianRpIjo
iZWMzM2NjYjctNjk2MC00MmQ1LWJmNjItZGU5NTc5NDQzMjg4In0.bakizOU_hXgIAlqN_U-
UZoOLW06gsphRqxNIsWoR4IeY6Wk-ZmZ7SYPLhVxsOzZxX-
A6NVCj7fbxgmpkZAeWdkYSQj53GtFvLrXyfyu0gjxcOLqk9yg-V7DQkTzro_R9bDEe9p-x5BX2Sky-
06gZQbPI23OZiwB-
7BL4F0Y15cwgmgA1mE1e_i96okg2Y3BkzNbOTkD1jpkgKRjws6bW77AyOL96e6LyM2b_Yo5XyAcp1Za-PG-
r2cEjS7Xbz0N4uvRtHuvFt5uB8fa4YnrF_5e2eBRPfrgnbqpjXwLHD7MA1nWd2FEyKRqhHafkj5Z6yeyhXVpYQeS
s2Db63efRGA

{
 "active": false
}
```

401 Unauthorized

Let dus op: een invalide client_assertion JWT token geeft een 401 Unauthorized terug, een invalide JWT token geeft een 200 OK met een {"active" : false} response terug.

Links naar gerelateerde onderwerpen

{}