Referentie & solution architectuur (WIP)

Nuts SSIBAC

Een solution architectuur voor het implementeren van ABAC gebasseerd op attributen verkregen via SSI. Maakt gebruik van OpenID4VP, DIDs, VCs, FHIR en OPA.

Introductie

Voor succesvolle gegevens uitwisseling in de zorg zijn een heel aantal (technische) stappen nodig. Deze stappen zijn afhankelijk van afspraken, standaarden en technology. Veel van deze afhankelijkheden vallen onder de generieke functies. Dit document gaat over het autoriseren van een gegevens vraag bij een FHIR endpoint.

Het Nuts initiatief promoot SSI als authenticatie oplossing binnen de zorg. Credentials en attributen verkregen via SSI hebben een hoog betrouwbaarheidsniveau en zijn daardoor uitermate geschikt voor gebruik in Attribute Based Access Control. Voor gegevensuitwisseling in de zorg is het van groot belang dat er hergebruik plaats vindt van technologie en standaarden over meerdere use cases. Als er uitgegaan wordt van authenticatie via SSI en het beschikbaar stellen van medische gegevens via een FHIR API, dan is er een autorisatie & accountability model nodig die de twee met elkaar verbindt. Dit document beschrijft zo'n model in de vorm van een referentie architectuur en geeft een voorbeeld van een implementatie mbv Open Source software.

Security model

plaatje lb -> authn -> authz -> forward

Access control policy

De basis van autorisatie is een autorisatievraag die door een access control policy (kort: policy) beantwoord moet worden. Zo'n policy is specifiek voor een use case. Verschillende use cases stellen verschillende eisen aan een policy. Een policy is afhankelijk van één of meerdere input parameters (kort: input). We beschrijven hier mogelijke input, niet elke policy maakt altijd gebruik van alle input.

Assertions

Bij het gebruik van SSI zullen er één of meerdere assertions vrijgegeven zijn door de gebruiker. Het access token dat afgegeven is door de autorisatie server is gekoppeld aan deze assertions. Bij de introspectie van het access token in de "authn" stap wordt het access token ingewisseld voor een lijst van attributen. Deze attributen kunnen als input gebruikt worden.

Request

Dit document beperkt zich tot het beschrijven van autorisaties van REST calls. De vraag naar een resource gaat middels een HTTP request (GET /fhir/Patient/4 HTTP/2.0). Deze informatie kan gebruikt worden in de policy.

Externe data

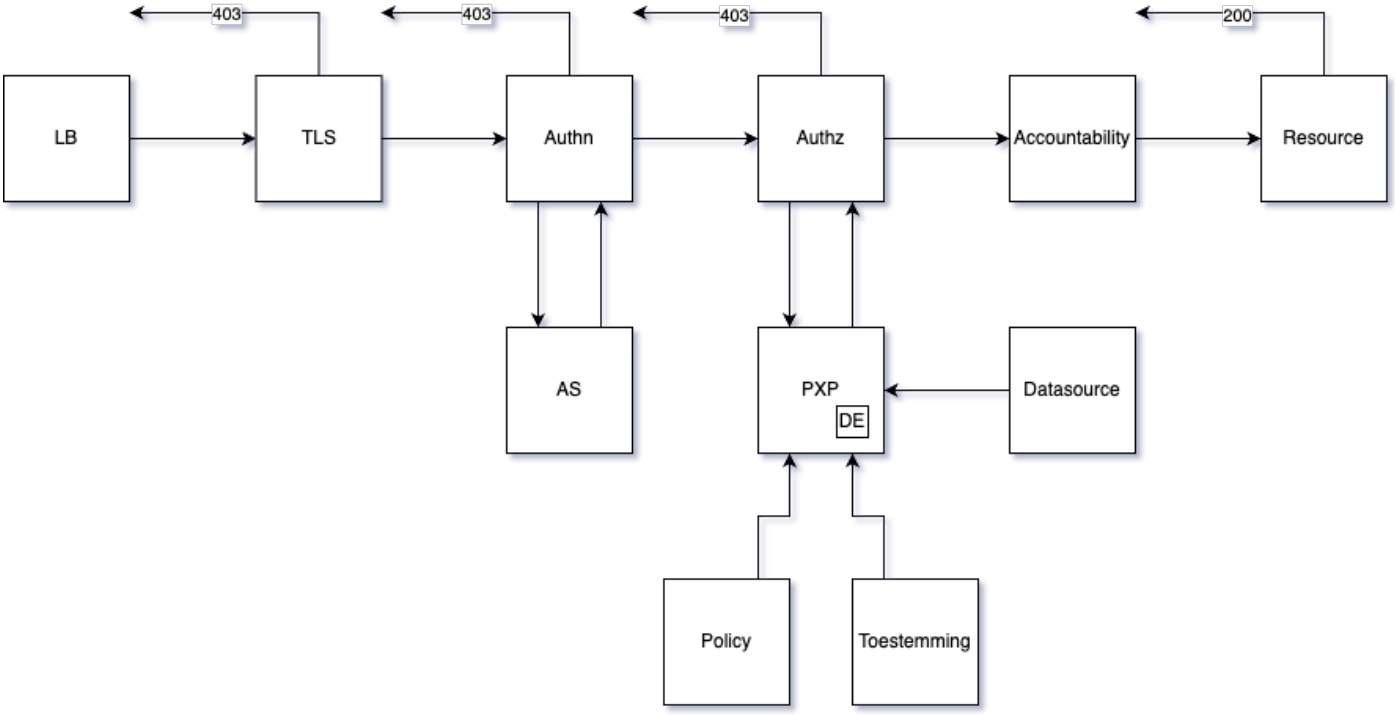
Indien de VC attributen en het request niet voldoende input is om tot een besluit te komen, kan er een afhankelijkheid zijn naar externe data.

Toestemming

In een use case staat beschreven wat de wettelijke grondslag is tot uitwisseling. Indien dit uitdrukkelijke toestemming is dan is deze toestemming input voor de policy.

SSIBAC referentiearchitectuur

Onderstaand model geeft de referentiearchitectuur weer voor SSIBAC.



Een uitleg van de verschillende componenten:

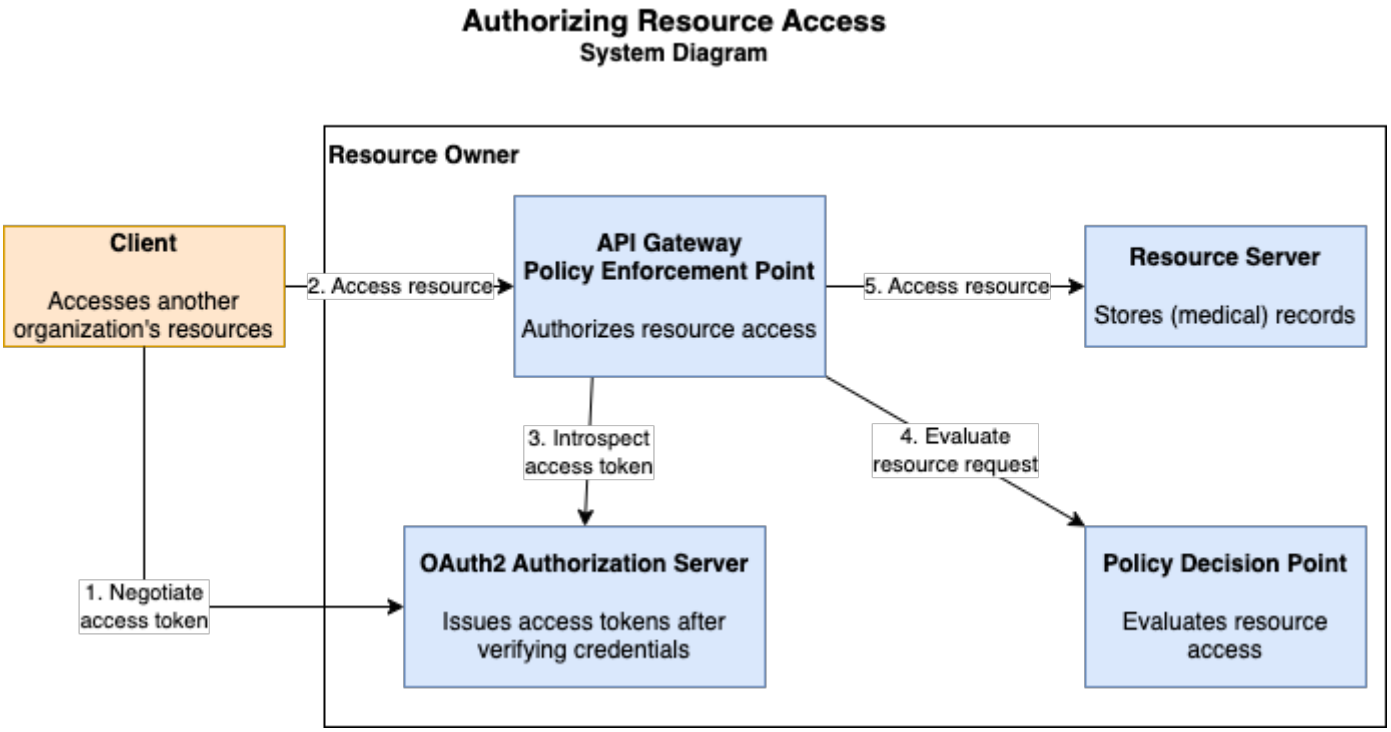
- **LB** - L4 Loadbalancer, verdeelt het verkeer over meerdere machines.
- **TLS** - TLS termination point. Het versleutelde kanaal gaat vanaf hier onversleuteld verder. Eventueel mTLS voor extra authenticatie.
- **Authn** - Controle van access token. Access token wordt bij Authorization Server (AS) ingewisseld voor assertions. Bij ongeldig access token stopt de flow.
- **AS** - Authorization server als uitgever van access tokens.
- **Authz** - Effectueren van autorisatie beslissing. Roept PXP aan voor autorisatiebeslissing. De flow stopt indien niet succesvol.
- **PXP** - Autorisatie component. Verzamelt de benodigde input en maakt een beslissing obv een policy. **DE** is de descision engine voor de policy.
- **Policy** - Getructureerde policy die door de DE uitgevoerd kan worden.
- **Toestemming** - Bron van toestemmingen.
- **Datasource** - Bron voor additionele input voor de PXP.
- **Accountability** - Bij positieve beslissing door PXP zorgt dit component voor de logging van het request.
- **Resource** - Database of service die de gevraagde data oplevert.

Het PXP component is weergegeven als centraal component dat gebruik maakt van meerdere input. Hoe het component toegang heeft tot de input is onderdeel van de solution architectuur.

Authorizing Resource Access

This section describes the steps involved to authorize resource access.

- The *client* proves eligibility for the use case (step 1 in the diagram below). This typically involves presenting a Verifiable Credential that proves the client is a registered care organization. It could include a use case-specific credential, acquired through an audit of the client's software and/or process.
- The *Policy Decision Point* checks whether the client is allowed to access the requested resources (step 4).



Use case afspraken

In het kader van herbruikbaarheid is het belangrijk dat alle use cases dezelfde standaarden gebruiken. Zie ook de keuzes van de Nuts SSIBAC solution architecture.

Per use case moet er afgesproken worden welke VCs/assertions er vanuit het authn component komen en welke input er gebruikt wordt.

Relatie tot XACML rollen

In de referentie architectuur worden de PAP, PDP en PIP samengevoegd tot PXP. De authn en authz componenten uitafbeelding de referentie architectuur vervullen samen de PEP rol.

Nuts SSIBAC solution architectuur

Hieronder volgen de keuzes van de Nuts SSIBAC solution architectuur die invulling geven aan de referentie architectuur.

Authenticatie

Voor het verkrijgen van een access token wordt OAuth2 en OpenID4VP gebruikt. De scope parameter in het OAuth2 autorisatie request is per use case gemapt op een Presentation Definition (kort: PD). Deze PD wordt gebruikt in de OpenID4VP flow om de juiste Verifiable Presentation te genereren door de client/wallet. Obv de VP geeft de autorisatie server een access token af.

De client voegt het access token toe aan het request via de Authorization http header. Het authn component stuurt de waarde van de header naar de autorisatie server. De AS geeft een introspection result terug volgens RFC7662. In dit resultaat staan iig de volgende attributen: `valid`, `client_id`, `sub` en `scope`. Daarnaast staan er key/value paren in waarbij de keys afkomstig zijn van de PD en de values de bijbehorende waardes uit de gestuurde VCs. De PD en de key/value paren moeten per use case vastgelegd worden in een Scope to Presentation Definition mapping. Alle waarden uit het introspection result kunnen gebruikt worden in de policy.

Het authn component stuurt het introspection result door naar het authz component via de `X-Userinfo` HTTP header.

Autorisatie

Het autorisatie component roept het PXP component aan met als input:

- **request** - de request line (eg: GET /fhir/patient/5 http/2.0)
- **X-Userinfo** - het resultaat van de token introspection (JSON)

Het PXP component antwoord met een JSON object met een enkele waarde `allow` (boolean). Op basis van dit antwoord stuurt het autorisatie component het request door of geeft het een 403 terug.

Het PXP component combineert zowel het maken van de beslissing als het opslaan van externe data. Door het combineren van deze twee rollen kunnen performance problemen vermeden worden indien er sprake is van grote hoeveelheden externe data. Het PXP component wordt geconfigureerd met een externe DB en de benodigde policy files. Nuts gebruik OPA als policy descision engine. Voor OPA zijn policy files geschreven in Rego. De policy files zijn gekoppeld aan de OAuth2 scope.

Het PXP component heeft 2 API's, 1 voor het toevoegen en verwijderen van externa data en 1 voor het uitvoeren van een beslissing. Het data model voor de externe data staat vast en ziet er als volgt uit:

```
{
"scope": "the_scope",
"client_id": "did or functional ID for the 'other' side",
"verifier_id": "your tenant ID",
"auth_input": {} //use case specific model
}
```

Ongeacht de implementatie van het authz en PXP component dient dit gestandaardiseerd te worden omdat dit tevens gebruikt wordt in OPA.

Bij het evalueren van een autorisatie vraag zal het PXP component de gegevens uit de token introspectie gebruiken (sub uit introspectie matcht op verifier_id) om valide `auth_input` te zoeken.

Voor het evalueren van een autorisatie vraag via OPA wordt de volgende input gebruikt:

- **input.request.method** - bevat de request method van het request
- **input.request.path** - bevat het pad uit het request, inclusief request params
- **input.<VAR>** - omvat alle variabelen uit het introspectie resultaat. Dus bijvoorbeeld **input.scope** voor de scope
- **input.external** - bevat alle gegevens onder `auth_input` uit de PXP database. Dit kan een union zijn van alle beschikbare data

Accountability

TODO

Bijlage I - voorbeelden

eOverdracht

For example, given the eOverdracht use case:

- Client requests an access token with scope `eOverdracht-receiver` (step 1)
  - Client proves:
    - to be a registered care organization by presenting `NutsOrganizationCredential`. This also identifies the organization (name and location)
    - to be participant in the eOverdracht use case by presenting `eOverdrachtCredential` (fictional)

- Client resource access of eOverdracht resources (e.g. FHIR resource `/Composition/<XYZ>`) is checked in a data store of registered eOverdracht flows:
  - "is client (identified by DID `did:web:example.com`) the receiver of eOverdracht `<ABC>`"
  - "is client requesting one of the resources associated with the eOverdracht flow `<ABC>`"'
  - "is client using an allowed HTTP method"
  - "is client using an allowed OAuth2 scope (`eOverdracht-receiver`)"

Bijlage II - Implementations

Policy Enforcement Point

There are several options available for implementing the PEP. Some are part of an API Gateway product that are able to route/load balance incoming requests, monitor traffic and secure endpoints. This section lists the available options.

API Gateways

- **Kong Enterprise** is a commercial product that supports Open Policy Agent, DPoP, and OAuth2 Token Introspection.
- **APISIX** is a free open-source product that supports Open Policy Agent, OAuth2 Token Introspection. It doesn't support DPoP.
  - An example setup with APISIX and OPA can be found here.
- **NGINX Plus** is a commercial product that supports OAuth2 Token Introspection.

Others

- **Nuts Policy Enforcement Point** is a free, open-source PEP that can be embedded in the Resource Server (as Golang library), or as standalone reverse proxy using Docker. It supports DPoP and OAuth2 Token Introspection.
- **NGINX**: It's possible to build OAuth2 Token Introspection, DPoP and Open Policy Agent into NGINX using JavaScript (a.k.a. *njs*).

Refs

https://www.dpss.inesc-id.pt/~mpc/pubs/SSIBAC__Self_Sovereign_Identity_Based_Access_Control.pdf

---

Revision #3
Created 31 May 2024 07:56:39 by Wout Slakhorst
Updated 3 September 2024 13:06:11 by Rein Krul