Process inrichting en Workflow van Authenticatie bij de NMD3.0

Versie: 0.1.1

Datum: 9/1/2019

Inleiding.

In dit document geven we eerst een korte uitleg bij de methodiek voor de Authenticatie bij de API van NMD3.0, en vooral ook de keuzes die daarbij gemaakt zijn, en waar die op gebaseerd zijn

Daarna volgt een beschrijving van het feitelijke authenticatie proces en Workflow, en tenslotte wordt ingegaan het verkrijgen van de benodigde refreshTokens voor dat authenticatieproces. Ook de beveiliging en de mogelijke risico's van het proces komen aan de orde.

Achtergrond: hoe zit het proces tussen klant / user, rekenbureau en NMD in elkaar?

Om te bedenken hoe de authenticatie en autorisatie voor de NMD3.0 en de applicaties en user interfaces die de rekenbureau maken in elkaar moeten zitten, moeten we natuurlijk zicht hebben op de onderliggende structuren en processflows, en eigendoms en verantwoordelijkheidsrelaties.

Op basis daarvan kunnen dan rationele keuzen worden gemaakt mbt de te gebruiken authenticatie en autorisatie

Daarom beginnen we met een simpel schema van de relatie tussen het rekenbureau en de klant / user enerzijds en tussen het rekenbureau en de NMD anderszijds.

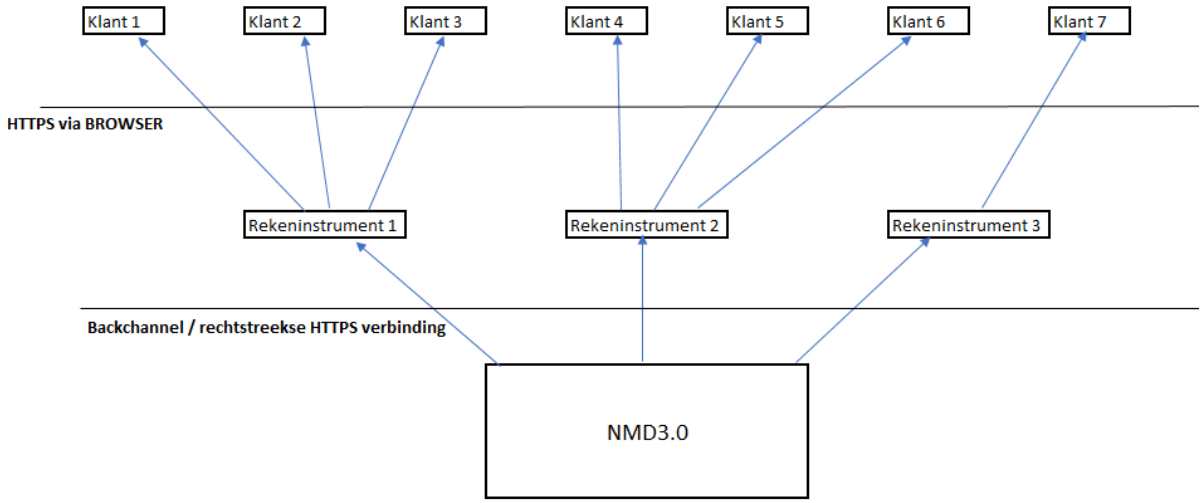

Fig. 1. De gekozen relatie tussen klanten / users, rekenbureaus en NMD

Wanneer we dit schema bekijken zien we dat er eigenlijk twee belangrijke stromen van data zijn:

1. Het dataverkeer tussen de klant van de instrumenthouder en diens applicatie op de eigen webserver
2. Het data verkeer tussen de webserver van de instrumenthouder en de webserver van NDM3.0

3. Kenmerkend aan dit schema is dat er pijlen lopen tussen de rekenbureaus en hun klanten / users en tussen de rekenbureaus en de NMD, maar **geen** pijlen rechtstreeks tussen de klant/user en de NMD. Dit is een bewuste keuze, die we hieronder zullen verantwoorden.

Keuze van een authenticatie mechanisme: Is Oauth2 van toepassing?

Wij denken niet dat Oauth2 in de standaardvorm van toepassing is, we leggen uit waarom, en welke versie we wel gebruiken.

Oauth2 wordt tegenwoordig als de standaard gezien voor authenticatie van dataverkeer in webservices en API's. De (meest gebruikte) standaardstructuur van Oauth2 wordt weergegeven in de illustratie hieronder:.

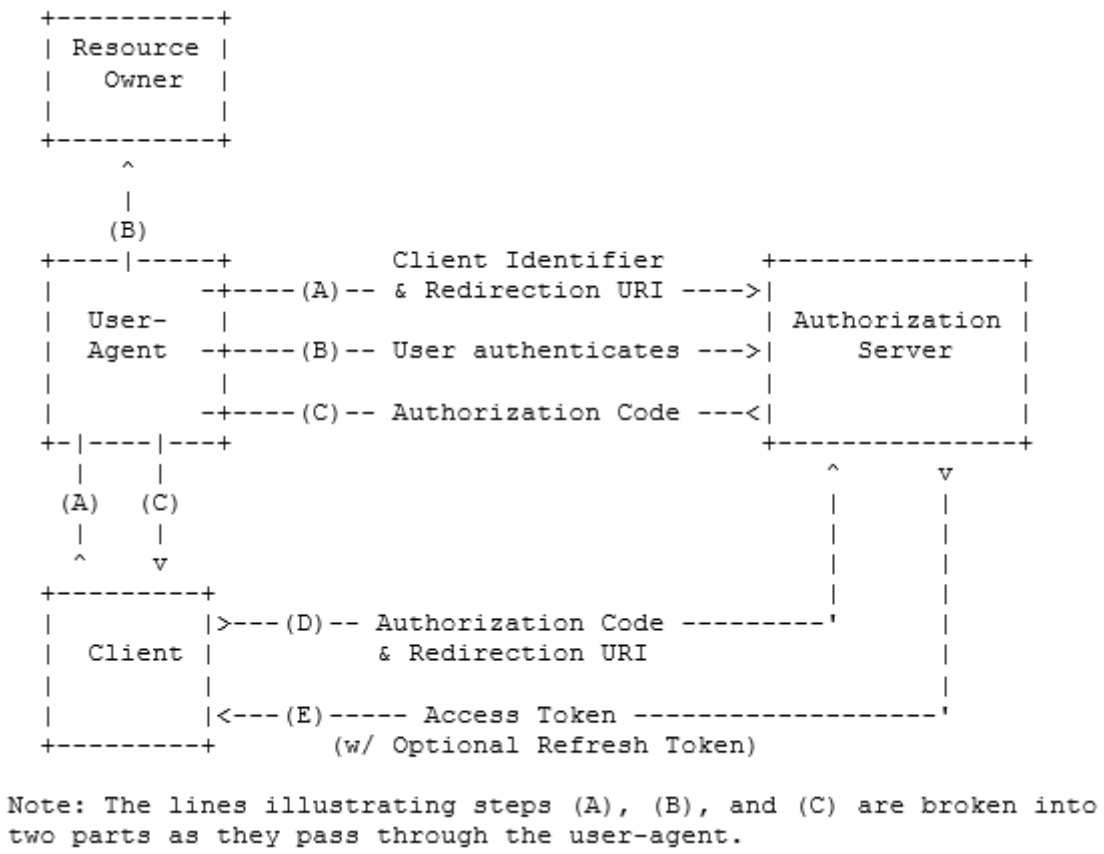

Fig. 2: Authorisation Code flow volgens par 4.1. van RFC 6749 van de Internet Engineering Task Force (IETF)

We zullen niet in de detail ingang op de processflows die hier worden weergegeven, die mogen onder de lezers van genoegzaam bekend worden verondersteld. Het punt waar het om gaat is de use-case die met deze min of meer standaard implementatie van Oauth2 wordt gefaciliteerd.

Kenmerkend aan dit proces is dat de autorisatie voor het gebruik van de gegevens van de user / resource owner door een app, client of webservice wordt gedelegeerd aan een authorisation server, die, na authenticatie en autorisatie door de user een autorisatie code en vervolgens een Access_Token verstrekt waarmee de client uiteindelijk het voorgestelde proces op de data kan uitvoeren.

Het probleem dat met deze constructie wordt opgelost is dat voor de komst hiervan de user credentials moest overdragen aan de client om deze in staat te stellen zich te authenticeren en authoriseren voor het gebruik van de users gegevens, waardoor die client vaak een onnodig grote bevoegdheid op een onnodig grote scope van de data verkreeg. Dit wordt opgelost door de delegatie van de autorisatie van het gegevensgebruik naar een aparte server of dienst.

Is de standaard architectuur van Oauth2 zinvol voor de API van NMD3.0?

De vraag is nu of standaard architectuur van Oauth2 zinvol is voor de relatie tussen de NMD, rekenbureau en de klanten / users van die laatste. Onze stelling is dat dit niet het geval is, om meerdere redenen.

De proces architectuur in het hierboven geschetste schema van Oauth2 zou, bij toepassing op de API van de NMD3.0, impliceren dat de user (die een klant is van het rekenbureau) met behulp van een client die door het rekenbureau aangeboden bewerkingen uitvoert op de data van de NMD, en zich daartoe authenticeert via de server van het rekenbureau waar hij op geadministreerd is, en die dan dus als authorisatie server dient.

In termen het schema uit fig.1. hierboven: we zouden dan ook lijnen moeten zien lopen rechtstreeks van de klant / user naar de NMD, met zijtakken naar de rekenbureaus voor de authenticatie.

Een dergelijk schema is hier NIET van toepassing!

Dit om meerdere redenen.

- De gegevens **uit de NMD** waarmee gerekend wordt zijn geen eigendom van de user. De NMD is in dit geval de resource owner, het is dus ook niet de user maar de NMD die autorisatie verleent om die gegevens te gebruiken.
- De user gebruikt de client niet om een dienst van de NMD te consumeren, maar om een dienst van het rekenbureau te consumeren.
- De rekenbureaus zijn onafhankelijke instelling die een eigen dienst verlenen aan hun klanten
- bovenstaande heeft alleen betrekking op de gegevens uit de NMD die gebruikt worden: het is natuurlijk best mogelijk dat er in de relatie tussen de user en het rekenbureau ook nog andere gegevens – bijvoorbeeld voor personalisatie, etc. –van de user worden opgeslagen bij de rekentool. Dit is echter een vraagstuk dat te maken heeft met de eigen authenticatie en autorisatie flow tussen de rekentool en de user, en daar staat NMD verder buiten (zie ook hieronder)

Daarnaast zijn er ook redenen waarom de procesarchitectuur zoals die door Oauth2 wordt geïmpliceerd minder gewenst is

1. **Inhoudelijke onafhankelijkheid van de rekenbureaus**. Wanneer SBK (als uitgever van de NMD) agnostisch blijft van de specifieke diensten die de rekenbureau aan hun klanten aanbieden hoeft SBK ook geen verantwoordelijkheid te nemen voor de arrangementen mbt soorten diensten, autorisaties tot gebruik daarvan etc etc. Dit vergroot sterk de vrijheid van het rekenbureau om zelf te bepalen welke diensten worden aangeboden, en hoe dit gebeurt
2. **Technische simplificatie.** De door ons hieronder voorgestelde koppelingswijze impliceert een zeer losse koppeling waarbij alleen datasets worden opgevraagd en aangeleverd, en er verder geen enkel verband is tussen de wederzijds processen. Dit maakt ook de risico's kleiner doordat de relaties zo simpel mogelijk blijven.

Samenvattend kunnen we stellen dat het model van Oauth2 **niet van toepassing** is op de relaties tussen users / klanten, rekenbureaus en NMD3.0

De vraag is natuurlijk: welk model willen we wel gebruiken?

Een lossere koppeling, de software van het rekenbureau authenticeert zich zelfstandig.

Gegevens bovenstaande factoren kiezen wij voor een oplossing waarin de circuits gescheiden blijven, in overeenstemming met fig.1. hierboven:

1. De user meldt zich aan bij de app of service van het rekenbureau, en wordt door het rekenbureau geauthenticeerd en geauthoriseerd
2. Wanneer de user / klant eenmaal met de user interface aan het werk is om de benodigde selecties te maken en berekeningen uit te voeren heeft het systeem van het rekenbureau natuurlijk data nodig om deze te faciliteren.
3. Wanneer dit het geval is meldt de software op de server van het rekenbureau zich op eigen account van het rekenbureau aan bij de API van de NMD3.0.
4. Dit initiatief tot aanmelding om data op te halen wordt door de software van het rekenbureau genomen op momenten dat dit, gegeven de interne procesflow opportuun is. Hierbij komt natuurlijk geen menselijk autorisatie van pas aangezien dit het werkproces van de user ernstig zou verstoren
5. Tussen het rekenbureau en de NMD ontstaat dus een autonome authenticatie, autorisatie en data flow.
6. Voor een dergelijke server to server “backchannel” communicatie bestaat ook een Oauth2 protocol, zoals is te vinden in paragraaf 4.4. van RFC 6749, mbt. De zogeheten “client credentials grant”. Deze hebben wij gebruikt als basis voor de inrichting van de authenticatie voor de server tot server communicatie tussen rekenbureau en NMD. Het schema hiervan is op de volgende pagina weergegeven
7. Maar let op: Verantwoordelijkheden zijn niet voor 100% gescheiden. Weliswaar gaan we er van uit dat de software van het rekenbureau zelfstandig en op eigen verantwoordelijkheid van het rekenbureau de API aanspreekt, de noodzaak om dit te doen wordt natuurlijk wel gecreëerd door de handelingen van de user, en de user krijgt daarna ook beschikking over de opgehaalde en gepresenteerde data. Dit betekent dat misbruik van de gegevens uit de NMD via de user interface van het rekenbureau toch tot op zekere hoogte mogelijk blijft. SBK zal aan de licentieovereenkomst op dit punt toevoegen dat een normaal te verwachten niveau van beveiliging bij de licentienemer aan de orde is op het punt van het niet kunnen muteren van NMD data en het niet zonder autorisatie kunnen aanroepen van de API..

De betreffende paragraaf uit RFC 6749:

Hieronder is de betreffende paragraaf uit RFC 6749 weergegeven waarin het mechanisme van de client_credentials grant wordt beschreven

4.4. Client Credentials Grant

The client can request an access token using only its client credentials (or other supported means of authentication) when the client is requesting access to the protected resources under its control, or those of another resource owner that have been previously arranged with the authorization server (the method of which is beyond the scope of this specification).

The client credentials grant type MUST only be used by confidential clients.

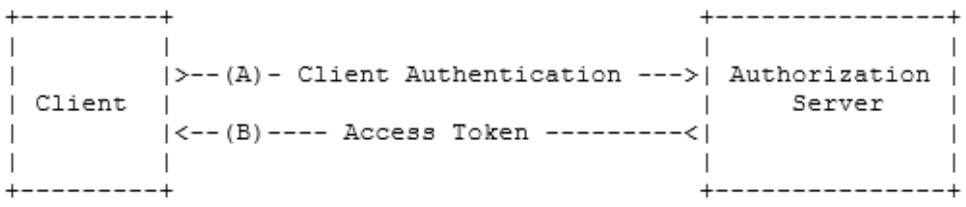

Figure 6: Client Credentials Flow

The flow illustrated in Figure 6 includes the following steps:

(A) The client authenticates with the authorization server and requests an access token from the token endpoint.

(B) The authorization server authenticates the client, and if valid, issues an access token.

4.4.1. Authorization Request and Response

Since the client authentication is used as the authorization grant, no additional authorization request is needed.

4.4.2. Access Token Request

The client makes a request to the token endpoint by adding the following parameters using the "application/x-www-form-urlencoded" format per Appendix B with a character encoding of UTF-8 in the HTTP request entity-body:

grant_type
: REQUIRED. Value MUST be set to "client_credentials".

scope
: OPTIONAL. The scope of the access request as described by Section 3.3.

The client MUST authenticate with the authorization server as described in Section 3.2.1.

Fig.3. Specificatie en illustratie van het proces van de client credentials grant.

Schema en Workflow van de Authenticatie van het proces van het rekenbureau bij de NMD3.0

Onderstaand schema geeft de workflow voor de authenticatie van de User bij het rekenbureau en van het rekenbureau bij de NMD weer. In dit schema speelt het gedeelte boven de horizontale lijn zich af tussen user / klant en rekenbureau, terwijl het gedeelte onder de horizontale lijn zich afspeelt tussen rekenbureau en NMD

LET OP: de user authenticatie tussen klant en rekenbureau is hier zeer schematisch weergegeven, omdat dit feitelijk los staat van de NMD, anders dan dat er natuurlijk wel goed beheerst authenticatieproces van de user moet zijn…

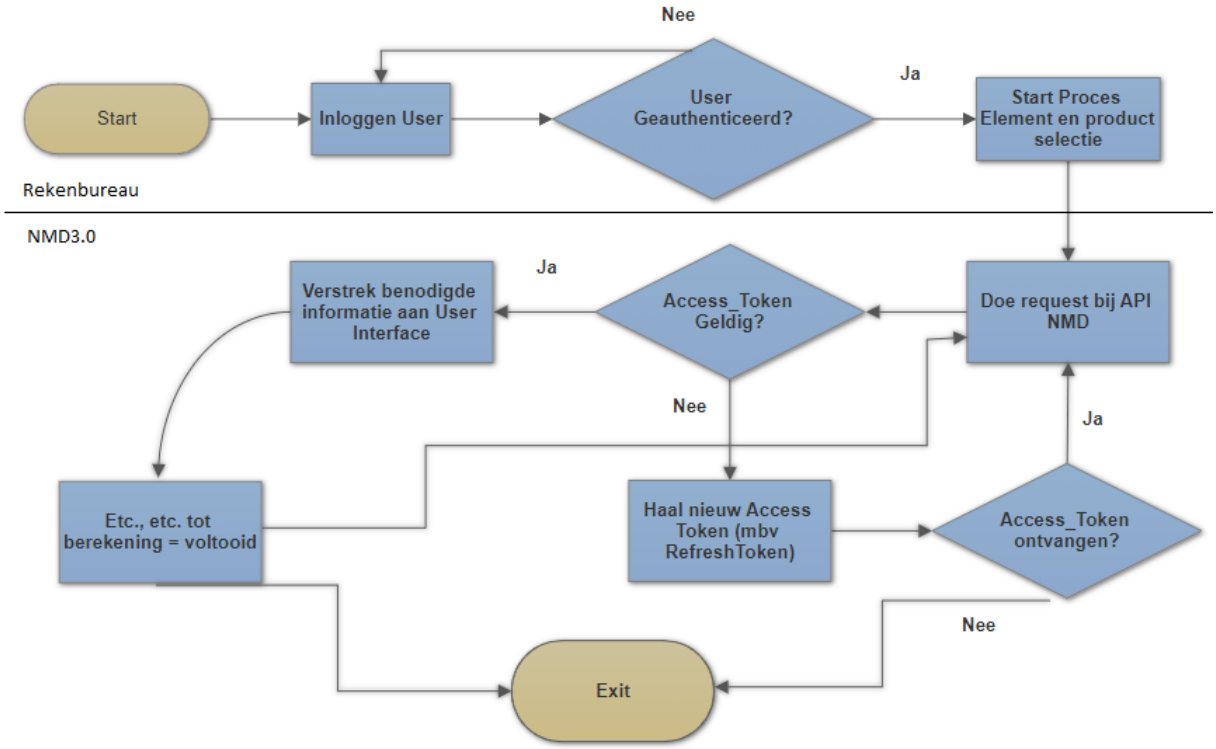

Fig.4. De procesflow van het authenticeren en ophalen van gegevens bij de NMD door het rekenbureau

Hierboven zien we nogmaals dat het niet de user is die zich aanmeld bij de NMD, maar de (software van) het rekenbureau die zich - nadat de user zich bij hen heeft geauthenticeerd - als zelfstandig entiteit en namens het Account van het rekenbureau aanmeldt en inlogt.

Het proces zoals beschreven in de RFC 6749 betreft daarbij het deelproces van de aanvraag van een nieuw Access_Token waarbij de beslissing op “Access_Token = geldig” met “Nee” wordt geantwoord. Hierbij dient een door SBK uit te geven “refreshToken” als client credential voor het proces van uitgifte van de Access_Token.

Voor het overige beschrijft het schema een vrij standaard proces flow van authenticeren mbv van een Access_Token, runnen van diverse requests voor informatie die het proces van de client nodig heeft om gewenste user interface op te bouwen, waar nodig nieuwe request opbouwen, bij iedere request opnieuw authenticeren met het Access_Token, etc. etc.

Workflow verkrijgen Access_Token:

(voor details: zie de technische documentatie / Readme verkrijgen Acces_Token)

1. Om het proces te starten meldt de applicatie van het rekenbureau zich met een normale informatie-request (bijvoorbeeld: AllReferenceObjects, volledige URI's worden natuurlijk verstrekt!) aan bij de API van de NMD, met in de header een bestaand Access_Token, of een willekeurige placeholder string wanneer er nog geen Access_Token aanwezig is.
2. Tijdens de verwerking van de request checkt de API van de NMD3.0 als eerste of het Access_Token correct en nog geldig is.
3. Wanneer het Access_Token inderdaad aan die voorwaarden voldoet wordt de opgevraagde informatie aangeleverd, en kan de normale flow van de applicatie worden voorgezet
4. Wanneer het Access_Token niet geldig of verlopen is retourneert de API JSON met daarin de message "Access_TokenExpired"
5. Bij een verlopen Access_Token meldt de applicatie zich aan bij de getToken Resource van de authenticatieserver, met in de headers het refreshToken, en het ID van de specifieke API waarvoor men Access_Token wordt aangevraagt (voorlopig: "1"), en in de request body de melding "grant_type=client_credentials", zoals hierboven beschreven
6. Het refreshToken en de overige informatie wordt door de AuthenticatieServer gecheckt, en als alles in orde is wordt JSON geretourneerd met in daar in het veld "TOKEN" als waarde het nieuwe Access_Token dat vervolgens in het proces gebruikt kan worden voor volgende logins, totdat het Access_Token weer verlopen is, en en weer een nieuw moet worden aangevraagd.
7. Voorlopig staat de geldigheidsduur van het Access_Token op 1 uur

Het refreshToken:

- Wordt verstrekt door SBK
- Tijdens de **ontwikkelperiode** is dit een handmatig aangemaakt Token, voor ieder rekenbureau wordt een apart tijdelijk refreshToken aangemaakt dat is gekoppeld aan het AccountID van het betreffende rekenbureau in de datastructuur (die wel al compleet is opgezet)
- In de **productie fase** zal dit een door het systeem aangemaakt Token zijn (op basis van de accountgegevens van het rekenbureau, dat door SBK wordt uitgegeven, en beheerd)
- Aan het beheer van de refreshTokens

Een belangrijk aspect is het **beheer van de refreshTokens**. Dit is onderdeel van de arrangementen met SBK en de ontwikkeling van het beheerssysteem die nog in volle gang is. Wordt vervolgd.

Risico analyse van de Server to server Communicatie voor deze structur.

1. De hele communicatie vindt plaats op basis van een geëncrypteerde https:// verbinding
2. Door het verwijderen van de extra proces stappen voor een apart autorisatie voor het gegevens gebruik zijn er ook minder complexiteiten en risicopunten waar foute implementaties problemen kunnen veroorzaken
3. De refreshToken en Access_Tokens worden aangemaakt met Microsoft Rijndael AES managed security op basis van een client secret en random salt, en onder andere een Timestamp, en nog een aantal andere gegevens
4. Het belangrijkste risico ligt daarbij eigenlijk op de server / in de software van de client /c.q. het rekenbureau.

- Het is belangrijk dat de clientsoftware het servercertificaat van de https:// verbinding goed valideert, anders blijven man in the middle attacks alsnog mogelijk
- Op de server van client is er het risico dat een programmeur of iemand met hoge toegangsrechten bijvoorbeeld het refreshToken steelt. In dit verband is zorgvuldig TokenBeheer noodzakelijk, en zullen ook refreshTokens met enige regelmaat (mogelijk eens in de 3 maanden) worden vervangen
- De implementatie van een clientsided X509 certificaat zou deze risico's kunnen verwijderen. Om dit echt goed doen zou deze implementatie dan echter ook client IP adres en het gecertificeerde pad daarvan moeten bevatten. Dit is voor een client behoorlijk ingewikkeld en stelt forse eisen aan de client en de beheerder daarvan.
- Een andere optie is om te werken met een HSM ofwel Hardware Security Module. Ook de implementatie daarvan is echter niet zonder problemen en kosten.

5. De nog onbeantwoorde vraag is of we beide bovengenoemde risicos's als dermate ernstig beoordelen dat het nodig wordt om ook op de client server een X509 certificaat of HSM te vereisen. Vooralnog lijkt ons dit niet nodig, omdat we te maken hebben met een zeer klein aantal (nu: 4!) vertrouwde partijen, waardoor goed beheer van de refreshTokens + monitoring van de communicatie voldoende zou moeten zijn.

Kort samengevat: dit hele server to server proces is goed afgeschermd, door het gebruik van HTTPS met bijbehorende server certificaat op de server van de NMD, en goed versleutelde Tokens. Het belangrijkste risico bevindt zich eigenlijk aan de kant van de rekenbureaus, waar een kwaadwillende met toegang tot de server en voldoende kennis, zoals een programmeur het refreshToken zou kunnen stelen, indien e.e.a. niet goed beveiligd is.

We houden om die reden nog in overweging om ook aan de kant van het rekenbureau een X509 certificaat te vereisen. Voorlopig zien we dit echter als niet nodig.