Whitebox token authenticatie in Topicus HAP

Om de integratie van Whitebox token authenticatie in Topicus HAP mogelijk te maken hebben wij een manier bedacht om de huidige token authenticatie ook praktisch te kunnen gebruiken binnen een webapplicatie draaiende in een browser. Hierin worden de Whitebox authenticatie tokens ondertekend middels ZorgID. Om de inspanning van het implementatie traject aan de kant van Topicus HAP te beperken hebben wij een voorbeeld implementatie geschreven in Kotlin. Deze kan gedeeltelijk gebruikt worden as is, of als voorbeeld dienen voor de daadwerkelijke implementatie.

Onderdelen

Dit project bevat twee onderdelen

1. WBXAuthToken class (WBXAuthtoken.kt)

De WBXAuthToken class representeert een authenticatie token zoals de Whitebox deze ondersteund.
De class stelt methodes beschikbaar om een authenticatie token te coderen tot een formaat dat de ZorgID SDK kan ondertekenen en een token in het juiste formaat (JWS) kan formateren zodat een Whitebox deze kan gebruiken/valideren.

2. Voorbeeld applicatie (Main.kt)

De voorbeeld applicatie demonstreert het proces van het genereren en verwerken van een authenticatie token.

WBXAuthToken class API

De constructor van de WBXAuthToken class neemt 3 argumenten: de Whitebox URL, de bijbehorende challenge en het UZI certificaat waarmee het token ondertekend gaat worden.

Daarnaast heeft de WBXAuthToken class 6 publieke methodes die het proces vormgeven.

1) updateTimestamp

Met deze methode wordt de timestamp van het token (de timestamp bepaald de uiteindelijke geldigheid van het token) naar de huidige datum en tijd ingesteld. Deze methode wordt aangeroepen bij het initialiseren van het token. Het is daarom niet nodig om deze zelf aan te roepen. Het is echter wel mogelijk (en aan te raden) om de timestamp te updaten indien er veel tijd zit tussen het

initialiseren van het token en het daadwerkelijk ondertekenen van het token. **Let op!** als de data voor ondertekening is opgehaald met de getHashBag methode moet de updateTimestamp methode niet meer worden aangeroepen. De timestamp is onderdeel van de data die wordt ondertekend dus als deze methode wordt aangeroepen na ondertekening wordt de uiteindelijk gegenereerde JWS ongeldig. Dit wordt momenteel niet geforceerd in de class zelf.

2) getHashBag

Deze methode geeft de data terug die ondertekend moet worden, gewrapped in een *SignDataRequestBag*. Deze *SignDataRequestBag* kan aan de sign methode van de transaction manager (onderdeel van de ZorgID SDK) worden meegeven om te worden ondertekend.

3) convertAndSetSignature

Deze methode converteert de handtekening naar het juiste formaat en slaat deze op voor later gebruik.
De handtekening die terug gegeven wordt vanuit de ZorgID SDK is gecodeerd in standaard base64 formaat. In de standaard die de Whitebox authenicatie token implementeerd (JWS) wordt een URL base64 encoding scheme gebruikt zonder padding.

4) jws

Deze methode geeft het Whitebox authenticatie token terug in het JWS compact formaat. Dit is het formaat dat de Whitebox verwacht.

5) url

Deze methode geeft de Whitebox URL terug waarvoor deze Whitebox authenticatie token toegang geeft.

6) endpoint

Deze methode geeft het endpoint (URL) terug waarnaar het token en url moeten worden opgestuurd (d.m.v een HTTP POST request).

Whitebox token authenticatie proces

Naast de TLS sessie authenticatie, die momenteel gebruikt wordt door Topicus HAP, biedt de Whitebox ook een token authenticatie methode aan. Deze methode is op een relatief simple manier te implementeren binnen Topicus HAP met behulp van de ZorgID SDK/applicatie en de WBXAuthToken class.

Om het proces te laten werken is er naast alleen de Whitebox URL (zoals in de huidige implementatie) ook een zogenaamde challenge nodig. Deze wordt verkregen bij het opvragen van de

Whitebox URL bij de Hapbox. Het betreffende veld is toegevoegd aan de search-and-copy call, die nu de volgende signature heeft:

Request:

```
POST /haplink/topicus/0.2/search-and-copy

{"bsn": "123443210"}
```

Response:

```
200 Ok

{
 "url": "https://vrbld-amsterdam.mcs-net.nl/a6516we98r4w9e8/HL7v3-PS",
 "challenge": "4257+pGn6EDeoiHYGKckFQ"
}
```

De challenge wordt samen met de URL en UZI pas gegevens ondertekend volgens de (JWS standaard).

Het proces is als volgt:

1. Initialiseer een WBXAuthToken met de URL, challenge en UZI certificaat van de huidige sessie
2. Roep de getHashBag methode aan
3. Roep de ITransactionManager.sign (ZorgID SDK) methode aan met de waarde die de getHashBag methode terug geeft
4. Wacht op de callback van de transaction manager (ZorgID SDK)
5. Extraheer de ondertekende data (eventArgs.signedDataBag.content[0].data)
6. Roep de convertAndSetSignature methode aan met de ondertekende data

Nu is het WBXAuthToken gereed om een dossier bij de Whitebox op te halen.

Om het dossier op te halen moet er een HTTP POST request wordt gedaan vanuit een iFrame met de volgende velden:

```
url=[de Whitebox url]
x-access-token=[de jws]
```

Voorbeeld van de content die in het iFrame moet worden geladen:

```
<html>
    <head></head>
    <body>
        <form method='POST' action='${wbxAuthToken.endpoint()}'>
            <input type='hidden' name='x-access-token' value='${wbxAuthToken.jws()}'>
            <input type='hidden' name='url' value='${wbxAuthToken.url()}'>
        </form>
        <script>document.querySelector('form').submit()</script>
    </body>
</html>
```

Er zijn verschillende manieren denkbaar om bovenstaande HTML in een iframe te krijgen:

1. genereer een data url van de inhoud van het iframe en geeft deze mee als het *src* attribuut
2. geef de inhoud van de iFrame mee het *srcdoc* attribuut
3. render de inhoud van het iFrame op een specifiek endpoint binnen Topicus HAP en geeft dit op als het *src* attribuut van het iFrame

Methode 1 en 2 werken alleen als de CSP van de pagina waarop de iFrame wordt geladen dit toelaat.

De pagina die wordt geladen na het POST verzoek zal, net als in de huidige implementatie, een message posten op zijn parent window met daarin de hoogte van de content zodat de parent window daarop het iFrame kan resizen.

Voorbeeld message

```
{
    'height': 500
}
```