Nieuwe basis voor beveiliging in SharePoint

Windows Identity Foundation biedt nieuwe mogelijkheden

Michiel van Otegem

Beveiliging is het laatste decennium veel complexer geworden. Niet alleen ontwikkelingen in technologie, maar ook social engineering zorgen ervoor dat we op een heel andere manier naar beveiliging moeten kijken. Windows Identity Foundation is een nieuwe basis voor beveiliging die bestaande problemen verhelpt en nieuwe mogelijkheden biedt. SharePoint 2010 is het eerste serverproduct dat hiervan gebruik zal maken, maar het werkt ook in Windows Azure en je kunt er zelf applicaties mee ontwikkelen.

Beveiliging lijkt zo simpel. Je ontwikkelt een applicatie, zorgt ervoor dat gebruikers een gebruikersnaam en wachtwoord krijgen, je zorgt dat een gebruiker alleen datgene kan waartoe hij/zij gerechtigd is en je bent klaar. Helaas heeft de gemiddelde gebruiker te maken met wel meer dan één applicatie en zo rijzen het aantal gebruikersnamen en wachtwoorden de pan uit. Veilig? Waarschijnlijk niet, want de gebruikers worden haast wel gedwongen om gebruikersnamen en wachtwoorden te gebruiken die ze eenvoudig kunnen onthouden. Een systeem waarbij de gebruiker iedere maand zijn/haar wachtwoord aan moet passen gaat daar niet bij helpen. Sterker nog, de kans is groot dat dit het probleem alleen maar erger maakt; in plaats van één ingewikkeld wachtwoord iedere maand een makkelijker wachtwoord. Nu is een unieke gebruikersnaam (met wachtwoord) voor sommige applicaties een noodzaak, maar zelfs als we het aantal gebruikersnamen dat iemand heeft kunnen verminderen, is het eigenlijk nog maar de vraag of het nodig is dat iedere applicatie je (hele) identiteit kent. Voor onze bedrijfsportal is het bijvoorbeeld meestal niet heel interessant om te weten wie ik ben, zolang het maar vast kan stellen dat ik de bedrijfsportal mag gebruiken (met andere woorden of ik medewerker ben) en of ik wel de rechten heb om de gegevens die ik probeer in te zien ook daadwerkelijk mag zien. Misschien dat het voor logging interessant is om te weten wie ik ben, maar functioneel gezien is dit niet noodzakelijk. Toch moet ik inloggen met mijn gebruikersnaam en wachtwoord, ook al gaat dat in dit geval automatisch, omdat onze portal zich in hetzelfde domein bevindt. In dit soort scenario's geldt ook nog eens dat de applicatie in principe toegang krijgt tot alle gegevens die over mij beschikbaar zijn in Active Directory. Binnen mijn bedrijf is dat niet zo erg, maar het is ten eerste overbodig en ten tweede niet wenselijk in een privacy gevoelige omgeving.

Claims Based Authorization

Het bovenstaande schetst in het kort de fundamentele problemen die we vandaag de dag hebben als het gaat om identiteit en beveiliging op basis daarvan. Er zijn nog meer problemen die zich voordoen, zoals bijvoorbeeld gebruikers van een andere organisatie die in business-to-business scenario's toegevoegd moeten worden aan de eigen userstore, maar deze vallen buiten deze introductie tot WIF. In essentie zorgt WIF ervoor dat de gebruiker één identiteit heeft en dat een systeem alleen gegevens over de identiteit krijgt die het nodig heeft. Deze twee eenvoudige eigenschappen scheppen de voorwaarden om de eerder geschetste problemen te verhelpen. De basis van WIF is zogenaamde Claims Based Authorization (CBA). De concepten van CBA zijn van essentieel belang om WIF goed toe te kunnen passen in je (SharePoint) applicaties. Bij CBA bestaat je identiteit uit een verzameling claims.

Een claim is een stukje informatie over jou, zoals je geboortedatum, het bedrijf waarvoor je werkt, enzovoorts. Het heet een claim en niet een feit, omdat een claim niet per definitie een echt feit over je hoeft te zijn. Je kunt dit zien wanneer je een Live ID gaat maken. Mits nog beschikbaar, kan iemand best een Live ID maken met de naam Jan-Peter Balkenende, ook al heet de gebruiker in het echt mevrouw Jansen. Wanneer mevrouw Jansen aangehouden wordt door de politie kan ze echter niet blijven volhouden dat ze Jan-Peter Balkenende is. Dit is waar de uitgever van de claim in beeld komt. Je rijbewijs wordt uitgegeven door de Nederlandse staat en we gaan ervan uit dat de informatie op je rijbewijs waarheidsgetrouw is. Wanneer Mevrouw Jansen haar rijbewijs laat zien, zal deze dus onomstotelijk bewijzen dat ze niet Jan-Peter Balkenende is. Mevrouw Jansen kan echter wel bij het aanmelden bij de sportschool een paar jaartjes van haar leeftijd afsnoepen. Het is daarom van belang dat de ontvanger van een claim de uitgever van de claim vertrouwt of eigenlijk vertrouwt dat de informatie accuraat is voor zover dat van belang is.

In het geval van het pasje van de sportschool is de uitgever van het pasje de sportschool zelf, dus de claim dat mevrouw Jansen lid is

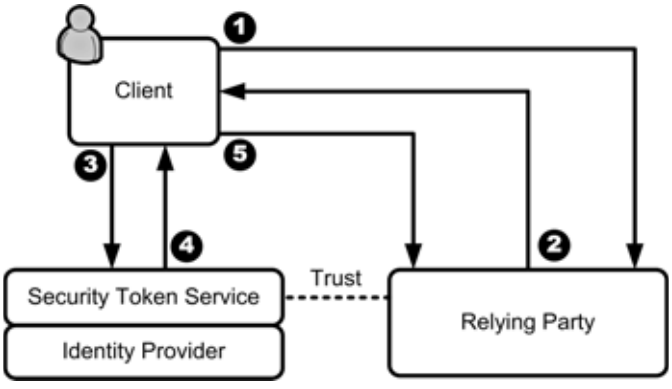
van de sportschool zit wel goed. De leeftijd zal de sportschool eigenlijk een worst wezen, dus dat die informatie niet accuraat is boeit niet echt. De politie moet er echter op kunnen vertrouwen dat gemeentes alleen waarheidsgetrouwe rijbewijzen afgeven. Een systeem in CBA dat claims accepteert noemen we daarom ook wel een Relying Party (RP), hetgeen Vertrouwende Partij betekent. Een RP kan een applicatie zijn, maar ook een webservice of andere resource. De RP vertrouwt een zogenaamde Issuer (uitgever) die we ook wel de Identity Provider (IP). Een voorbeeld van een IP is Active Directory of Live ID. CBA werkt op basis van tokens en een IP heeft derhalve een deel nodig dat tokens kan uitgeven. We noemen dit de Security Token Service (STS). Hou er rekening mee dat in veel documentatie IP en STS door elkaar heen gebruikt worden, ondanks dat een STS een maar een onderdeel is van een IP.

WIF werkt op basis van Security Assertion Markup Language (SAML) tokens

Een STS geeft tokens uit op basis van een policy. De policy bepaalt welke claims een RP krijgt. Omdat er een vertrouwensrelatie (Trust) nodig is tussen de RP en de STS, dient een RP zich te registreren bij de STS. De RP geeft daarbij met de policy aan welke claims de RP nodig heeft. Het is aan de beheerder van de STS om dit goed te keuren. Dit is belangrijk, want daarmee wordt gewaarborgd dat een RP alleen die claims krijgt die het nodig heeft. De RP zal mijn adres niet krijgen als alleen mijn naam en leeftijd voldoende zijn en daardoor ben ik zeker dat mijn privacy niet geschonden wordt. Wanneer een RP geregistreerd is bij een STS die ik gebruik, kan ik gebruikmaken van deze RP. Figuur 1 laat zien hoe het proces (in theorie) verloopt wanneer iemand een RP gebruikt. Hierin vinden de volgende stappen plaats: de client vraagt toegang tot de applicatie (1), de applicatie retourneert de policy aan de client (2), de client meldt zich aan bij de STS (indien nog nodig) en vraagt om een token met de juiste policy (3), de STS retourneert een token met de juiste policy (4), die de client vervolgens aan de applicatie aanbiedt (5).

WIF in de praktijk

Met de theorie achter de rug kunnen we kijken hoe WIF in de praktijk werkt. Om met WIF te werken moet je het eerst installeren. Het is een aparte download die werkt op Windows Vista/ Windows 2008 Server en hoger met .NET Framework 3.5 of ho-



FIGUUR 1: AUTHENTICATIEPROCES MET CLAIMS BASED AUTHORIZATION.

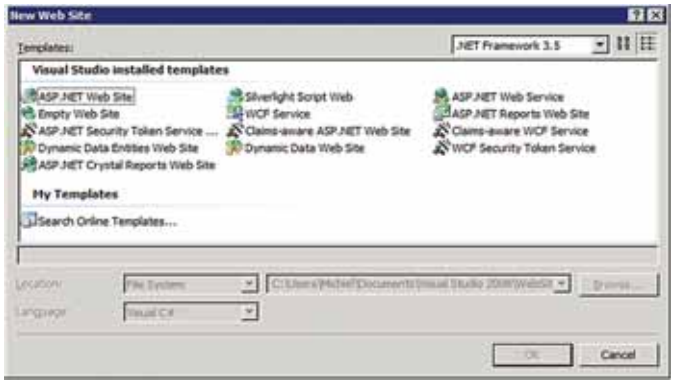
IN HET GEVAL VAN HET PASJE VAN DE SPORTSCHOOL IS DE UITGEVER VAN HET PASJE DE SPORTSCHOOL ZELF, DUS DE CLAIM DAT MEVROUW JANSEN LID IS VAN DE SPORTSCHOOL ZIT WEL GOED.

ger. De download is beschikbaar via de Identity Management sectie van MSDN (http://msdn.microsoft.com/en-us/security/aa570351.aspx), waar ook informatie staat over Active Directory Federation Services 2.0 (ADFS) en Windows Cardspace 2.0 die samen met WIF ontwikkeld zijn. ADFS is een serverproduct van Microsoft dat dient als Identity Provider en waar een STS onderdeel van uitmaakt. Cardspace is het Identity Card systeem van Microsoft waarmee de gebruiker controle heeft over 'pasjes' die naar een RP gestuurd worden om toegang te krijgen. WIF werkt uiteraard uitstekend samen met deze technologieën, maar is daar niet toe beperkt. Omdat WIF werkt op basis van Security Assertion Markup Language (SAML) tokens, en de open specificaties WS-Trust en WS-Federation hoeft een met WIF gemaakte RP niet gebruik te maken van een STS die ook met WIF gemaakt is. Een STS die ontwikkeld is in bijvoorbeeld Java werkt ook prima, zolang de RP de tokens van die STS maar accepteert.

Voor ontwikkelen met WIF heb je naast WIF zelf ook de WIF SDK nodig. De SDK installeert een aantal nieuwe projecttemplates en add-ins in Visual Studio. De projecttemplates zijn beschikbaar via New > Web Site … en in figuur 2 zie je keuzes die je dan hebt. In totaal zijn er vier nieuwe templates, twee voor ASP.NET en twee voor WCF. Elk van deze twee opties biedt een STS en een Claims Aware website of service. Claims Aware betekent niets anders dan dat de website of service met claims kan werken. Wanneer je een Claims Aware website maakt, krijg je een hele simpele website met een default- en een login-pagina. Wat voor je geregeld is, is een referentie naar de Microsoft.IdentityModel assembly die het hart van WIF vormt, een web.config die klaar is voor configuratie van WIF, en wat code die claims uit kan lezen in de default-pagina. Op zich werkt deze 'applicatie', maar deze is ten eerste niet veilig en ten tweede maakt het nog geen gebruik van een externe Identity Provider. Je kunt hier verandering in brengen door de Login-pagina te verwijderen, zodat inloggen niet meer mogelijk is via de website zelf en vervolgens te verwijzen naar een STS. Dit doe je door met de rechter muisknop te klikken op de website en te kiezen voor Add STS reference… Je krijgt dan een scherm te zien met daarin een verwijzing naar de web.config van de website en de URL waaronder de website beschikbaar is. Als je deze onder HTTPS wilt hosten, wat voor productie-applicaties wel zo handig is, dan moet je hier een IIS URL opgeven. Voor ontwikkeldoeleinden is het prima om het zo te laten en de waarschuwing na het klikken op Next te negeren. Je krijgt dan het scherm in figuur 3 te zien.

Standaard is in figuur 3 de optie No STS geselecteerd en die is alleen nuttig als je een bestaande applicatie hebt waar je de nodige referenties in aan wilt brengen voor WIF. In dit geval is dat al zo en is dus alleen de vraag of we gebruik gaan maken van een nieu-

In metadata wordt beschreven welke claims de STS kan aanbieden en staat ook de informatie over het certificaat van de STS.



FIGUUR 2: WIF PROJECT TEMPLATES.

we of een bestaande STS. Aangezien je waarschijnlijk nog geen bestaande STS hebt, is het makkelijkste om er een te laten maken, zodat je hierin rond kunt gaan snuffelen. Als je wel een bestaande STS hebt, moet je verwijzen naar metadata die de STS beschrijft. In die metadata wordt beschreven welke claims de STS aan kan bieden en staat informatie over het certificaat van de STS, zodat de RP kan controleren of een token daadwerkelijk is uitgegeven door de STS. Dit bestand wordt automatisch aangemaakt als je een STS maakt met de wizard of wanneer je een STS project type kiest. Daarna zul je zelf dit bestand aan moeten passen als je claims toevoegt of verwijderd, of code schrijven die dit voor je doet. De STS die gemaakt wordt bevat ook weer een default- en een login-pagina. De login-pagina is voor de gebruiker om zich te authenticeren, maar die in de project template vereist alleen een naam en geen wachtwoord. Dat zul je zelf moeten verzorgen, bijvoorbeeld met ASP.NET Membership. De login-pagina verwijst naar de default-pagina die de token te maakt met de juiste claims. Het hart van die code zit in de GetOutputClaimsIdentity methode in CustomSecurityToken.cs. Hierin wordt voor de aangemelde principal een ClaimsIdentity gemaakt met een verzameling claims. Aangezien voor een ClaimsIdentity alleen claims van belang zijn, is dat alles wat je aan de ClaimsIdentity hoeft toe te voegen, zoals te zien in codevoorbeeld 1. Merk op dat in de code twee voorgedefinieerde claim types gebruikt worden en een die zelf gedefinieerd is. In principe is een claim type niet meer dan een URI verwijzing die de claim uniek maakt. OASIS heeft een standaard claim types en Microsoft heeft daar nog wat aan toegevoegd. Deze kun je invoegen met de ClaimTypes class. Houdt er rekening mee dat in .NET 3.5 al beperkte ondersteuning voor claims zit en dat dit geïmplementeerd is in de System.IdentityModel.Claims namespace. Met WIF gebruik je de Microsoft.IdentityModel.Claims namespace. Echter, in voorbeeldcode worden de ClaimTypes classes uit beide namespaces door elkaar heen gebruikt. Om verwarring te voorkomen raad ik aan om alleen de ClaimTypes te gebruiken in de WIF namespace, aangezien die alles bevat wat de oorspronkelijke class ook bevat en meer.

```
// Maak een ClaimsIdentity met naam, organisatie en email claims
ClaimsIdentity identity = new ClaimsIdentity();
identity.Claims.Add(new Claim(ClaimTypes.Name, principal.Identity.
Name));
identity.Claims.Add(new Claim("http://netmag/organization", "Bata-
viaLabs"));
identity.Claims.Add(new Claim(ClaimTypes.Email, "michiel@aspnl.com"));
```

CODEVOORBEELD 1.

Nadat de token is opgebouwd wordt de gebruiker verwezen naar de website waarvoor de gebruiker een token nodig had. De default-pagina in de standaard Claims Aware Website leest de token uit en toont welke claims erin zitten. Codevoorbeeld 2 is hier een simpele variant van die de belangrijkste eigenschappen van een claim laten zien, te weten het type, de waarde, het data type en de uitgever. Zoals je in codevoorbeeld 2 kunt zien kun je de principal en de identity uit de context (in dit geval de pagina) halen, zoals je dit met andere soorten principals en identities ook kunt doen. Je hoeft dus niet zo gek veel anders te doen om met claims te kunnen werken. Sterker nog, je kunt je bestaande applicaties betrekkelijk eenvoudig veranderen zodat ze claims gebruiken, door gebruik te maken van de zogenaamde Role-claim die Microsoft geïntroduceerd heeft. Een Role-claim is een claim die correspondeert met rollen (groepen) in een applicatie. Als je dus nu een applicatie hebt die beveiligt op basis van groepen, dan is het redelijk eenvoudig om die groepen uit de STS te laten komen en de applicatie met minimale wijzigingen te laten werken met WIF.

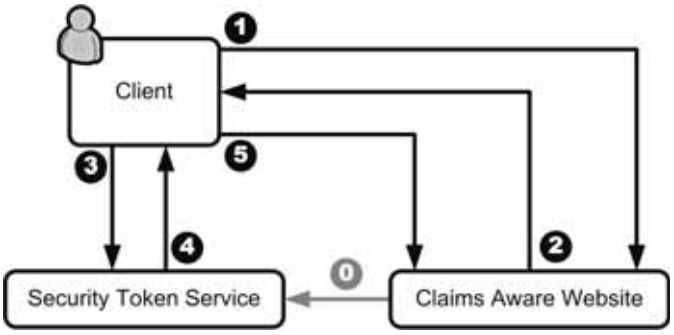
```
IClaimsPrincipal principal = Page.User as IClaimsPrincipal;
IClaimsIdentity identity = (IClaimsIdentity)principal.Identity;
foreach(Claim claim in identity.Claims)
{
    Response.Write(claim.ClaimType);
    Response.Write(claim.Value);
    Response.Write(claim.ValueType);
    Response.Write(claim.Issuer);

    Response.Write("<hr>");
}
```

CODEVOORBEELD 2.

Als je zowel de website als de STS doorneemt zul je zien dat de praktijk net even anders is dan de theorie. Figuur 4 laat zien hoe de praktijk van WIF er in dit scenario uitziet. Vooral stap 2 is daarbij anders, omdat de Claims Aware Website de gebruiker naar de STS doorstuurt, maar niet aangeeft welke policy het nodig heeft. De website is in stap 0, die niet door de gebruiker geïnitieerd wordt, geregistreerd bij de STS. Het standaard STS project stuurt ongeacht de RP een token met dezelfde claims naar de client, dus daarin moet de code nog aangebracht worden om te differentiëren per RP. Het idee is daarom dat je de STS aanpast om dit te kunnen en daarbij een management console of iets dergelijks te verzorgen waarmee een beheerder kan bepalen welke claims een applicatie ontvangt. Indien een organisatie van plan is

FIGUUR 3, STS BEHEER VOOR EEN WEBSITE OF SERVICE.



FIGUUR 4: WIF IN DE PRAKTIJK.

WIF in alle applicaties in te zetten, dan loont het de moeite om naar Active Directory Federation Services 2.0 te kijken, omdat die op basis van een Windows domein account tokens uit kan geven en het beheren van claims voor applicaties daar standaard al in zit.

Voor- en nadelen van WIF

WIF is niet meer en niet minder dan een implementatie van een verzameling standaarden die Claims Based Authorization mogelijk maken. Het is daardoor bij uitstek de technologie om te integreren met andere platformen en om Single Sign On te realiseren op het internet. Doordat het zo eenvoudig te gebruiken is in ASP.NET websites en WCF services staat er weinig in te weg om in ieder geval binnen de Microsoft-wereld CBA tot een groot succes te maken en het eigen domein, eigen applicaties en diensten van derden (bijvoorbeeld in Windows Azure) aan elkaar te koppelen. De openheid zorgt er echter voor dat ook andere vendors en platformen aan kunnen haken, dus als je nu je applicaties met WIF bouwt, dan is de kans groot dat je later eenvoudig aan andere applicaties en services kunt koppelen. Er zijn eigenlijk maar twee technologische aandachtspunten met betrekking tot CBA. Ten eerste is CBA maar zo sterk als de zwakste schakel, omdat CBA Single Sign On kan verzorgen. Als iemand een STS weet te kraken of het certificaat ervan weet na te bootsen, dan is het ecosysteem rond die STS onveilig, aangezien alle RPs die de STS (direct of indirect) vertrouwen dan toegankelijk zijn voor kwaadwillenden. Echter alleen binnen de mogelijkheden die de STS biedt op basis van de claims die de STS uitgeeft en de RP gebruikt. Ten tweede is het theoretisch mogelijk om een policy te maken waardoor een token een enorme hoeveelheid data met zich mee moet dragen. Dit is echter iets dat de ontwikkelaar van een applicatie of service expliciet moet doen. Daarnaast moet de beheerder van de STS of de gebruiker zelf (in het geval van Cardspace) de policy accepteren. De kans op problemen is hierdoor veel kleiner dan bijvoorbeeld bij de Microsoft-implementatie van Kerberos waarbij alle beveiligingsgroepen waar een gebruiker toe behoort meegestuurd worden in de token, ongeacht of dit nodig is of niet. Door claims handig in te delen hoeft een token over het algemeen niet echt groot te zijn om een applicatie te voorzien van de gegevens die het nodig heeft. Dit is ook vooral de uitdaging: het ontwerpen van de juiste policy voor een applicatie.

Michiel van Otegem, is directeur/chief software architect bij BataviaLabs