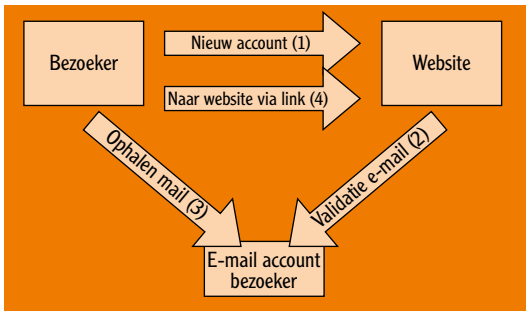
Enterprise

Je wilt gebruik maken van de diensten van een website. Hiervoor moet je eerst een nieuw account aanmaken. Je vult je e-mailadres en naam in en je moet een wachtwoord opgeven. Vervolgens word je verzocht je e-mailadres te valideren middels een speciale link. Met het volgen van deze link ben je weer terug op de site en je account is klaar voor gebruik.

OpenID: snel en veilig inloggen op internet

Verantwoordelijkheid meer bij bezoeker website



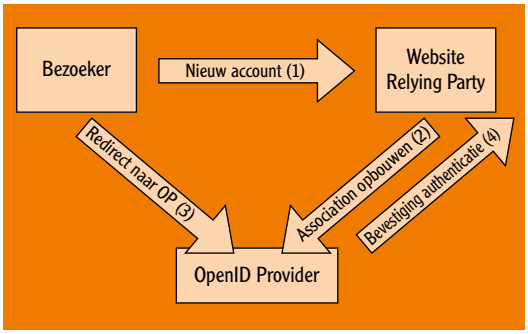
Het zojuist geschetste scenario is ongekend populair op het internet, en niet zonder reden. E-mail is uitgegroeid tot een laagdrempelig authenticatiemechanisme voor websites dat door iedereen wordt geaccepteerd. Alleen jammer dat we dit zo vaak moeten doen, steeds weer dezelfde nieuwe wachtwoorden onthouden, vaak dezelfde gegevens invullen in een net weer ander formaat…

Wie bekruipt niet het gevoel dat we dit beter kunnen regelen? Een veilige, snelle en makkelijke manier om op websites in te kunnen loggen, dat is het doel van OpenID.

Wat is OpenID?

OpenID is onstaan in 2005 en opgezet door Brad Fitzpatrick, de oprichter van de blogservice LiveJournal. Gebruikers van deze blogservice wilden zich niet steeds aanmelden op andere blogsites als ze toch al een LiveJournal sessie gestart waren. Dit is uitgegroeid tot de huidige OpenID standaard en de oprichting van de OpenID foundation, die het bewaken van de OpenID specificaties als doel heeft gesteld.

In december 2007 is de 2.0-versie van de OpenID specificatie opgesteld; in dit artikel wordt uitgegaan van deze versie van de specificatie.
OpenID is een protocol dat decentrale authenticatie op websites mogelijk maakt. Met OpenID kan iedereen zich identificeren op dezelfde manier zoals websites dat doen: met behulp van een URL.
Met dit protocol in de hand kan een website de authenticatie 'uitbesteden' aan een zogenaamde OpenID Provider (verder OP). Een website die de authenticatie op deze manier uitbesteedt wordt een Relying Party (verder RP) genoemd. OpenID schrijft hierbij niet voor hoe de authenticatie bij de OP precies plaatsvindt. De bezoeker van een website bepaalt zelf welke OP hij gebruikt om bij een RP in te loggen. De koppeling van een identiteit aan bijbehorende credentials is dus decentraal geregeld.
Op het moment dat je in het scenario uit de inleiding het e-mailadres vervangt door een URL, het e-mailaccount vervangt door een OP-account en de authenticatie door middel van een validatie-mail overlaat aan de RP en OP heb je het idee van OpenID al te pakken. Dit scenario is afgebeeld in de onderstaande figuur:



Bas Stoker
is Java consultant bij Info Support

Het bericht waarmee de OP de authenticatie bevestigt (stap 4) wordt binnen OpenID ook wel een Positive Assertion bericht genoemd. Dit bericht wordt verstuurd middels een redirect en hiermee is de gebruiker gelijk weer terug op de website van de RP.

Het OpenID-scenario heeft de volgende voordelen voor de betrokken partijen:
Voor de eindgebruiker:
- Minder verschillende wachtwoorden nodig
- Mogelijkheid om sneller in te loggen (met cookie van OP)
- Wachtwoord alléén bekend bij OP

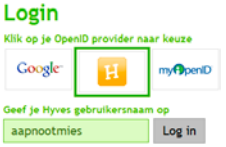
Voor Relying Party:
- Lagere registratiedrempel voor bezoekers
- Kostenbesparing op wachtwoordmanagement

Voor OpenID Provider:
- Meer traffic naar website

Inloggen met een URL?

Een OpenID URL waarmee je inlogt ziet er, afhankelijk van je provider, verschillend uit. Voorbeelden hiervan zijn aapnootmies.myopenid.com of www.google.com/accounts/o8/id.
Hierbij is het niet verplicht om een unieke OpenID-URL op te geven. Als een OP dit ondersteunt mag deze URL ook alleen verwijzen naar de OP zelf. Een voorbeeld hiervan is de wat vreemd uitziende URL van Google, die voor iedereen met een Google-account hetzelfde is.
Een Claimed ID is waar het uiteindelijk om draait. Dit is de unieke identificatie waaraan RP's bezoekers mogen herkennen. Dit is niet altijd gelijk aan de URL die een gebruiker opgeeft op de website van een RP. De specificatie maakt namelijk onderscheid tussen twee soorten URLs die door bezoekers opgegeven mogen worden: Provider ID's en End User ID's.
Dit is het beste uit te leggen met een inlogscherm van een voorbeeld-RP:

De URL http://aapnootmies.eenprovider.nl uit het getoonde inlogscherm is een voorbeeld van een End User ID.
Om het een bezoeker makkelijk te maken zijn er inmiddels veel websites die buttons opnemen met logo's van populaire OpenID Providers:



De Google button is niets meer dan een shortcut voor de Provider ID van Google.
De buttons voor Hyves en myOpenID zijn dit niet, maar stellen de bezoeker in staat in te loggen met een gebruikersnaam i.p.v. een URL, zoals het schermvoorbeeld hierboven laat zien.
Als een bezoeker inlogt door middel van een Provider ID zal de OP vervolgens vragen om een gebruikersnaam. Hiermee komt de OP tot een unieke OpenID-URL, een zogenaamd 'OpenID Claimed ID'. Op het moment dat je inlogt door middel van een End User ID kan de teruggeven Claimed ID gelijk zijn aan de opgegeven End User ID. Het kan er ook van afwijken, zoals bijvoorbeeld bij Google en Hyves het geval is.

Provider	Gekozen of Opgegeven OpenID	Claimed ID	Type ID
Google	google.com/accounts/o8/id	www.google.com/accounts/o8/id?id=aitoafertwvp3(...)	Provider ID
Hyves	aapnootmies.hyves.nl	aapnootmies.hyves.nl/#NWYxYTY5OG	End User ID
myOpenID	aapnootmies.myopenid.nl	aapnootmies.myopenid.nl	End User ID
een provider	aapnootmies.eenprovider.nl	...	End User ID

Een Claimed ID wordt bij een succesvolle inlogpoging teruggestuurd naar de RP via een Positive Assertion bericht. De RP slaat dit ID meestal op in een database om een gebruiker bij een volgend bezoek weer te kunnen koppelen aan een bestaande gebruiker in de database.

Discovery

Discovery is het proces waarbij een RP met behulp van een OpenID URL een OP Endpoint URL achterhaalt. Dit is een absolute URL van een OP-server waarmee OpenID-berichten uitgewisseld kunnen worden. Discovery kan plaatsvinden via Yadis discovery of via een HTML discovery.
Yadis als specificatie staat los van OpenID, maar er wordt vanuit de OpenID-specificatie wel naar verwezen. Het gebruik van Yadis is verplicht gesteld voor discovery van een Provider ID. Yadis zelf ligt verder buiten de scope van dit artikel. Wel is het goed om te weten dat deze manier van discovery uiteindelijk leidt tot een OP Endpoint URL. HTML discovery vindt plaats doordat het HTML-document op de door de gebruiker opgegeven locatie bepaalde link-tags bevat:

```
<link rel="openid2.provider"
      href="http://www.eenprovider.nl/server"/>
```

Het href-attribuut van deze tag wijst hierbij ook altijd naar een OP Endpoint.

Association

Middels de discovery weet de RP hoe er contact kan worden opgenomen met de OP. De volgende stap is het opbouwen van een Association tussen de

RP en de OP. Deze associatie bestaat uit een uniek shared secret dat wordt uitgewisseld. De specificatie schrijft voor dat deze uitwisseling alleen mag plaatsvinden via encryptie op transportniveau (TLS) of via encryptie middels een sleutel die is uitgewisseld via een Diffie-Hellman sleuteluitwisseling. Deze associatie wordt door de RP gebruikt tijdens het verwerken van het Positive Assertion bericht. Het Positive Assertion bericht bevat een handtekening waarmee de afkomst met behulp van het shared secret gecontroleerd dient te worden. Een Association kan bij toekomstige verzoeken worden hergebruikt. De OP geeft bij een association ook aan hoelang deze geldig blijft.
Nadat de association tot stand is gebracht zal de RP de gebruiker doorsturen naar de website van de OP. Als de gebruiker zich bij de OP heeft weten te authenticeren (middels wachtwoord of reeds beschikbaar cookie) zal de OP de gebruiker weer terugsturen naar de website van de RP middels een Positive Assertion bericht.

OpenID4Java

Voor het Java-platform is er inmiddels een breed gedragen Open Source implementatie beschikbaar onder de naam OpenID4Java. Dit project is gehost op Google code en wordt met zo'n 18 committers goed gesupport. OpenID4java is een erg complete library; elk onderdeel van de specs heeft z'n plek gekregen en is ook goed terug te herkennen in de implementatie, mede door de duidelijke packagestructuur en goede documentatie. Er wordt ondersteuning geboden voor zowel de rol van RP als OP. Voor beide scenario's zijn ook voorbeelden meegeleverd die je als beginner snel op weg weten te helpen.

Voor het Java-platform is inmiddels een breed gedragen Open Source implementatie beschikbaar.

De meeste implementaties van OpenID zullen betrekking hebben op het ondersteunen van de rol van RP in een webapplicatie. De stappen die hiervoor nodig zijn lopen we in dit artikel dan ook door.

Om te beginnen kent OpenID4Java de class ConsumerManager. Deze class dankt zijn naam aan de oude OpenID 1.0-term voor Relying Party, Consumer. De ConsumerManager is het startpunt voor alle interactie met de library. Er is ook een aantal globale instellingen te wijzigen, voornamelijk op het gebied van security.

Het opnemen van de volgende code in een servlet is genoeg voor het versturen van een OpenID authenticatieverzoek naar een OP. In dit voorbeeld wordt uitgegaan van een door een bezoeker opgegeven End User ID met als parameternaam 'openid_identifier' en een beschikbare instantie van een ConsumerManager:

```
// 1) Voer de discovery stap uit
// met de meegegeven openid
String openid =
        request.getParameter("openid_identifier");
List<DiscoveryInformation> discoveries =
        consumerManager.discover(openid);

// 2) Probeer een association op te bouwen met OP
DiscoveryInformation discovered =
consumerManager.associate(discoveries);

// 3) Sla de discovery informatie op in de sessie
session.setAttribute("discovered", discovered);

// 4) Verkrijg een AuthRequest bericht object
AuthRequest authReq = consumerManager.authenticate(
discovered, openidReturnURL);

// 5) Maak authReq beschikbaar voor view
request.setAttribute("authReq", authReq);

// Genereer redirect-form met parameters uit authReq
```

Tijdens de associatiestap (2) vindt de directe communicatie plaats tussen de RP en de OP.

Er wordt met OpenID4Java ook een voorbeeld van een redirect-form meegeleverd. Met dit formulier wordt een aantal parameters naar de OP verstuurd. Deze parameters zijn afkomstig uit de AuthRequest instantie en bevat bij het inloggen met een End User ID onder andere de volgende variabelen:

```
openid.claimed_id = aapnootmies.eenprovider.nl
openid.return_to = http://modernewebsite.nl/
openidreturn/
openid.realm = modernewebsite.nl
openid.assoc_handle = djueuaij24kkakd
```

De variabele 'openid.claimed_id' bevat de opgegeven End User ID. De variabele 'openid.return_to' bevat de URL waarnaar de OP het Positive Assertion bericht verstuurt.
De variabele 'openid.realm' is de naam die de OP aan de gebruiker kan laten zien, om aan te geven welke website om het authenticatieverzoek heeft verzocht. De variabele 'openid.assoc_handle' bevat een verwijzing naar de eerder opgezette association, indien beschikbaar.
Het gebruik van een sessie om het DiscoveryInformation object op te slaan heeft de voorkeur, maar wordt vanuit de specificatie niet verplicht gesteld. Zonder associatie zal er in de volgende stap een extra roundtrip naar de OP nodig zijn om de geldigheid van de Positive Assertion bericht te valideren. Ook dit scenario wordt door OpenID4Java ondersteund. Dit scenario wordt in de OpenID-specificatie aangeduid als Stateless Verification.
Nadat de redirect heeft plaatsgevonden authenticeert de bezoeker van modernewebsite.nl zich bij zijn of haar OP. Bij een succesvolle authenticatie zal de OP een Positive Assertion bericht sturen naar de RP waarbij als URL de waarde van de 'openid.return_to' parameter wordt gebruikt.

De code om dit bericht te verwerken, ziet er als volgt uit:

```
// Haal de parameters uit het return-bericht van OP
ParameterList openidResp =
new ParameterList(request.getParameterMap());

// Haal de OpenID-specifieke state uit session
DiscoveryInformation discovered =
(DiscoveryInformation) session.
getAttribute("discovered");

// Controleer response
VerificationResult verification;
String receivingUrl = request.getRequestURL().
toString();
verification = consumerManager.verify(
receivingUrl, openidResp, discovered);
Identifier verified = verification.getVerifiedId();

if (verified != null) {
        // Ingelogd via OpenID!
        String openid = verified.getIdentifier();
} else {
        // Inlogpoging mislukt
}
```

De variabele 'openid' kan vervolgens op de sessie worden gezet en eventueel worden opgeslagen. Hiermee is de bezoeker ingelogd op de website en is het OpenID-scenario afgerond.

Attribute Exchange

Bovenop OpenID is er een aparte specificatie gedefinieerd genaamd Attribute Exchange. Deze specificatie definieert de manier waarop RP's extra attributen omtrent een identiteit kunnen opvragen bij de desbetreffende OP. Er is inmiddels een set van acht veelgebruikte attributen opgesteld die door veel OP's ondersteund worden. hieronder vallen onder andere een e-mailadres, naam, geslacht, tijdzone en een geboortedatum.

Typische Use-Case

Wat is naast sociale netwerksites en weblogs nu een goede Use-Case voor OpenID? Hierbij zou je kunnen denken aan een webwinkel. Potentiële klanten hoeven geen registratieproces meer te doorlopen, maar kunnen gelijk van start. Ook is de kans groter dat bezoekers terugkomen, omdat login gegevens minder makkelijk vergeten worden. Er zijn dus minder mensen die afhaken in het hele verkoopproces.

Voordelen ontwikkelaar

Het is voor een ontwikkelaar niet eenvoudig om een goed en veilig loginsysteem te bouwen voor een website. Daarnaast moeten er vaak Use-Cases gerealiseerd worden omtrent wachtwoordmanagement. Een belangrijk deel van dit werk kan met OpenID worden uitbesteed aan een OpenID Provider. Als een ontwikkelaar zich daarbij houdt aan de OpenID-specificaties (en dat is met een goede library niet zo veel werk meer) dan is de kans op security-lekken kleiner dan bij een eigen implementatie.

Acceptatie

Waarschijnlijk heb je, misschien zonder het te weten, al één of twee OpenID's op zak.
Sinds partijen als Hyves en Google zich opstellen als OpenID Provider is de gemiddelde Nederlander al aardig op weg. Maar er zijn nog weinig websites die zich opstellen als Relying Party. Hier en daar kan je met behulp van OpenID een stukje commentaar kwijt onderaan een blog, maar daar blijft het vaak nog bij. De tijd zal het leren of we binnenkort met ons OpenID, ook buiten de blogcommunity, kunnen inloggen op meer van onze favoriete websites.

Uitdagingen

De gebruikerservaring over websites heen is nog inconsistent. Door de verschillende opbouw van de OpenID-URL en verschillende combinaties van instellingen voor cookies op zowel RP- als OP-websites zijn voor een minder geïnformeerde bezoeker nog een bron van verrassingen. Hoe reageert een gebruiker de eerste keer wanneer er opeens is ingelogd zonder een wachtwoord op te geven? Is dat met de conclusie: "Ik had blijkbaar nog een sessie openstaan bij mijn OpenID provider"? De opties op dit gebied zullen nog moeten convergeren naar een 'standaard' zoals ook het e-mail/wachtwoord-scenario is ontstaan.
Een ander nadeel van OpenID is de gevoeligheid voor phishing. Omdat een redirect naar een OP deel uitmaakt van het scenario kan een website met minder goede bedoelingen hier doorverwijzen naar een zelf opgezette namaaksite. Een niets vermoedende bezoeker geeft hier vervolgens zijn wachtwoord op. Vanaf dat moment is het voor de eigenaar van de phishingsite een koud kunstje om dit gestolen OpenID uit te proberen op populaire websites die OpenID ondersteunen. Een belangrijke uitdaging op securitygebied ligt bij de manier waarop een bezoeker zich bij zijn OpenID Provider authenticeert. Gelukkig is een aantal OP's op dit gebied al aan het experimenteren met nieuwe methoden.

Conclusie

Het bouwen van een website met een veilig en gebruiksvriendelijk loginsysteem ligt met OpenID voor veel toepassingen binnen handbereik. Het is niet langer de website die bepaalt op welke manier een bezoeker zich authenticeert, maar de bezoeker zelf. Maar met deze 'power' heb je ook meer 'responsibility'. Is de gemiddelde bezoeker van de website hier klaar voor? Het antwoord hierop zal per website nog verschillen. «

De website bepaalt niet langer hoe een bezoeker zich aanmeldt.

Referenties:

http://openid.net/
http://www.openidentiteit.nl/
http://yadis.org/
http://code.google.com/p/openid4java/
http://groups.google.com/group/google-federated-login-api/