

**Claims Based Access Control
goed alternatief voor RBAC?**

**Trust management als basis
voor online vertrouwen**

**Behavioral information security:
attitude, kennis en gedrag bepalend**

**IBpedia is kennis maken én
kennis delen**

INFORMATIEBEVEILIGING

Claims based access control

Auteur: André Koot > André Koot is Security Manager bij Univé-VGZ-IZA-Trias. Hij is tevens hoofdredacteur van dit blad en is bereikbaar via a.koot@unive.nl.

Steeds meer belangrijke bedrijfsapplicaties worden als webtoepassingen in een Service Oriented Architectuur beschikbaar gesteld via het internet. Traditionele methoden voor toegangsbeveiliging zoals RBAC voldoen niet binnen dit architectuurprincipe en ze stellen de organisatie bovendien bloot aan significante beveiligingsrisico's. In dit artikel presenteer ik een alternatieve methode om webtoepassingen binnen een SOA beter te beveiligen.

Het afgelopen jaar zijn in dit blad diverse artikelen geschreven over het ontwikkelen en toepassen van Role Based Access Control. Ik start met het uitwerken van enkele bezwaren tegen het gebruik van RBAC in een Service Oriented Architecture. De reden hiervoor is dat we in de praktijk binnen een SOA omgeving diverse knelpunten tegenkomen op het gebied van toegangscontrole waar we met RBAC niet mee uit de voeten kunnen. Die knelpunten moeten wel worden gezien in het licht van het openstellen van een SOA voor toegang vanaf het internet, denk hierbij aan Web 2.0 toepassingen. Ik heb in een eerdere uitgave van dit blad al geschreven over het autoriseren in een Web 2.0 omgeving¹. Dit artikel bevat het vervolg hierop, met dien verstande dat ik het onderwerp iets verder heb uitgewerkt en dat het voorgestelde concept in de praktijk al eens beperkt is getoetst op toepasbaarheid. Daar staat tegenover dat het concept als geheel vermoedelijk niet zonder meer in de grootschalige praktijk uitvoerbaar is. Verschillende randvoorwaarden ontbreken nog, maar daar kom ik later op terug.

Ik laat eerst vier knelpunten van de traditionele access control methoden binnen een SOA en/of Web 2.0 omgeving de revue passeren.

1) RBAC werkt alleen als we identiteiten kunnen koppelen aan rollen

Dit is misschien wel de grootste beperking van RBAC. De methodiek impliceert enerzijds het koppelen van een set permissies binnen informatiesystemen aan een rol, anderzijds het koppelen van een identiteit aan diezelfde rol. Het beginsel is fraai, het betekent namelijk dat we niet

elke identiteit zelf hoeven te autoriseren voor alle objecten waarvoor het individu rechten moet hebben.

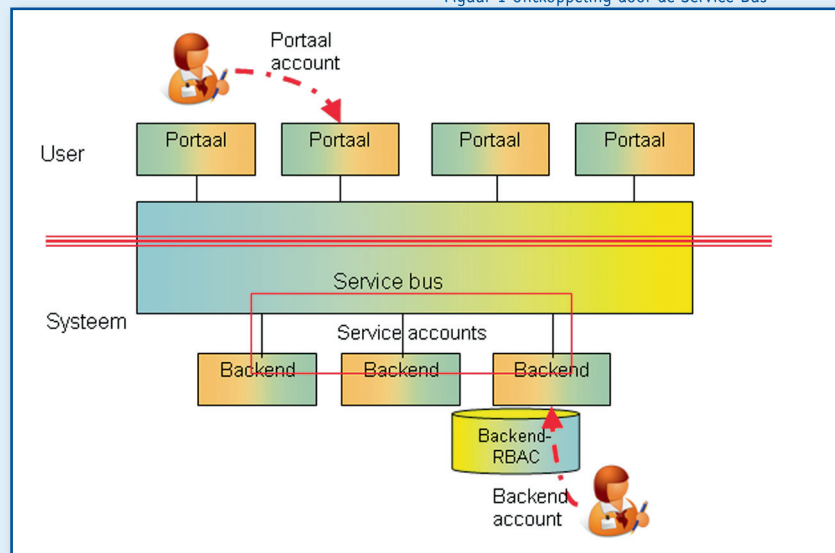
Het probleem is dat we, om dit principe te kunnen toepassen, wel de identiteiten moeten kennen, willen we ze kunnen koppelen aan een rol. We moeten in een autorisatiebeheersysteem een userid lid maken van een groep of rol. Maar wat als we geen userid hebben om te koppelen aan een rol? Dat lijkt misschien een overbodige opmerking, maar stel dat een medewerker vanaf het internet toegang zoekt tot een informatiesysteem. Dat is op zich simpel, de medewerker moet dan maar eerst inloggen. De verificatie van inloggen doen we in de Active Directory (AD) en de user krijgt de bij de rol behorende autorisaties. Maar in een webomgeving die aan een SOA is gekoppeld hebben we niet per definitie een AD. De AD bevindt zich in een veilige omgeving, niet in een DMZ of bij een hosting provider. Bovendien kunnen ook onbekende identiteiten zich via de website aanmelden. Denk aan klanten en partners.

Dat maakt al meteen duidelijk dat we niet een standaardrol kunnen gebruiken om anonieme internetgebruikers autorisaties te verschaffen. Een standaardrol is per definitie te beperkt. Het heeft geen zin om een businesspartner de standaardrol van klant te geven.

2) In een SOA heeft de identiteit geen autorisaties

Een tweede groot bezwaar tegen de traditionele toegangscontrole is dat binnen een SOA een identiteit zelf geen toegang heeft tot backendsystemen. Backendsystemen staan achter de servicebus. Het is de servicebus, of een gekoppelde processervice, die de achterliggende services aanstuurt. Een gebruikersidentiteit mag niet rechtstreeks services aanspreken. Een gebruikersidentiteit binnen het portaal mag dan ook geen autorisaties in het backendsysteem hebben. Dit zou een serieus beveiligingsrisico inhouden.

Figuur 1 Ontkoppeling door de Service Bus



[1] Verschenen in Informatiebeveiliging 2008 nummer 6

Wat betekent dat dan voor de rollen die iemand heeft? Niets. Het begrip rol komt in een SOA niet voor. Het RBAC systeem van een backendsysteem kent de identiteit van de portaalgebruiker niet en kan dan ook de autorisatie van een serviceaanroep niet valideren tegen een rol.

Er is een kleine uitweg: als je de identiteit (het userid met eventuele rollen) in een XML bericht kwijt kunt, dan zou een service op basis van de inhoud van een dergelijk bericht autorisatieregels kunnen toepassen. Maar dat doet enerzijds onrecht aan de eigenschappen van de webservice standaarden en anderzijds betekent dat aanpassen van de backendservices om het userid attribuut in een XML-bericht te kunnen herkennen. Als we zelf berichten gaan definiëren, dan zijn ook alleen die services te gebruiken die dergelijke aanpassingen aankunnen. Weg standaarden en minder mogelijkheid voor interoperabiliteit en integratie met externe (niet door ons beheerde) systemen.

Ontkoppeling van users en de backend systemen via de servicebus zorgt ervoor dat RBAC niet kan worden toegepast.

De systeemaccounts moeten de autorisaties hebben om alle taken van gebruikers in backendsystemen te kunnen uitvoeren. Maar dat is niet precies wat we willen, daarmee zijn we de Control volkomen kwijt. In de backendsystemen is immer het systeemaccount de eigenaar van alle transacties.

Ontkoppeling maakt het probleem misschien nog groter: je zou door in te breken in de SOA op een door een legitieme account geïnitieerde transactie kunnen meeliften en je kunnen voordoen als een legitieme gebruiker.

Dus moeten we voor de traceerbaarheid alleen al een compenserende maatregel treffen in de vorm van het realiseren van een audittrail. Ik kom daar later op terug, maar ik verwijs daarnaast ook naar mijn eerdere artikel over Audit Based Access Control².

3) RBAC heeft geen weet van de context van toegang

RBAC is een goed uitgekristalliseerd

concept. In essentie betekent RBAC dat elke extra rol ook extra autorisaties met zich meebrengt. Het toekennen van meer rollen leidt tot meer autorisaties. En daarbij zouden ook wel eens conflicterende autorisaties kunnen ontstaan.

Bijkomend probleem is dat RBAC geen rekening houdt met de situatie dat er, los van de rol, ook andere factoren invloed kunnen hebben op de effectieve autorisaties. Autorisaties kunnen naast rolgebaseerd ook regel-, content- of contextgebaseerd zijn. Denk bijvoorbeeld aan de situatie dat iemand nu eens niet via een kantoorwerkplek, maar via een laptop in de trein aan het werk gaat. Of na kantoortijd. Of via een privé pc. De betreffende medewerker heeft daarmee niet opeens een andere rol, maar de context waarbinnen de rol wordt uitgeoefend is wel verschillend. En dat kan betekenen dat de effectieve autorisaties wel eens anders zouden moeten zijn. Denk ook aan horizontale autorisaties, waarbij twee functionarissen weliswaar dezelfde functie en rol hebben, maar beiden slechts een deelverzameling van de gegevens mogen raadplegen. Dat kan voorkomen doordat ze beiden vanuit een andere juridische competentie actief zijn of binnen een organisatorische eenheid door afwijkende privacyregels en beperkingen de werkzaamheden separaat moeten uitvoeren: de ene persoon zou de gegevens, waarvoor de ander competent is, niet eens mogen zien. Beiden hebben wel dezelfde rol, maar feitelijk niet dezelfde autorisaties.

Waar een rol dus autorisaties toevoegt, zou de context de autorisaties moeten beperken. Maar, zoals gezegd, RBAC kan hier niet mee overweg. Er zouden aanvullende businessrules gedefinieerd moeten worden. 'Rule based access control' is dan wellicht de oplossing die daarvoor ingezet kan worden, maar daarmee krijg je een nieuw beheerprobleem.

4) RBAC kan niet overweg met eigenschappen van identiteiten

Wat te denken van de volgende casus.

Medewerker 1 is senior medewerker. Medewerker 2 is junior medewerker. Beiden hebben dezelfde rol. Je kunt je zo voorstellen dat RBAC daarmee niet uit de voeten kan. Je kunt nu wel de rol senior en de rol junior apart definiëren, maar het was nu juist de bedoeling van RBAC om het beheer van autorisaties te vereenvoudigen. Ook specialisaties van de ene functionaris ten opzichte van de andere kun je standaard niet onderkennen, tenzij je ook daar extra rollen definieert en de noodzakelijke autorisaties per competentie voor iedere rol apart instelt. In de praktijk zie je door dit probleem het aantal rollen exploderen. Role Mining is dan een oplossing om de essentie van de rollen te vinden, maar ieders functie is toch eigenlijk zo specifiek dat de generieke rollen een miskennen van de individuele capaciteiten zijn.

Conclusie ten aanzien van RBAC en SOA

De meeste van deze knelpunten zijn niet specifiek RBAC, of specifiek SOA of Web 2.0, maar we hebben er in een RBAC/SOA omgeving wel heel veel last van. RBAC en andere autorisatiemechanismen zijn preventieve maatregelen, er wordt mee beoogd om zo min mogelijk overschrijding van toegang te laten ontstaan en om daarmee op voorhand de beoogde functiescheiding te realiseren. Maar RBAC werkt niet goed binnen een SOA. Zonder additionele en compenserende maatregelen heeft in een SOA een service alle autorisaties. En iedereen die de betreffende service weet aan te spreken heeft daarmee dus ook alle autorisaties. In dat geval lopen we dus een aanzienlijk risico.

Zo simpel is het, we moeten op zoek naar een ander autorisatieparadigma om in control te komen. Tenminste, wanneer we portalen gebruiken om gebruikers toegang te geven tot backend systemen met een servicebus die zorgt voor het doorsnijden van de identiteit- / autorisatieketen.

Toegangscontrole

Misschien moeten we eerst het begrip toegangscontrole nog eens onder

[2] Informatiebeveiliging nummer 1 uit 2007

de loop nemen. Toegangscontrole is misschien niet de juiste term. In het Engels wordt de term Access Control niet voor niets gebruikt, het impliceert dat je de toegang beheerst. Niet zozeer dat je de toegang beperkt, maar eerder dat je verantwoording kunt nemen voor de feitelijk autorisaties. En dat kan betekenen dat je bewust te veel autorisaties toestaat, maar dat je wel weet wat er gebeurt. Als je weet waar je tekortschiet en de verantwoordelijkheid daarvoor kunt dragen, dan ben je wel in control.

Gesteld dat je het probleem onderkent, dan moet je nadenken over de oplossing van het probleem. Wij hebben die voor ons in ieder geval gevonden in een nieuwe ontwikkeling, namelijk het toepassen van Identity 2.0 mogelijkheden. Dat was althans voor ons de trigger om het RBAC probleem op een andere manier aan te pakken.

Identity 2.0

Identity 2.0 is het gedachtegoed om hergebruik van identiteiten mogelijk te maken. Een individu creëert of verkrijgt een eigen digitale identiteit en kan deze identiteit voor verschillende doeleinden toepassen. Het identity 2.0 concept maakt gebruik van verschillende open standaarden, waardoor het hergebruik van digitale identiteiten technisch en in de praktijk ook mogelijk wordt. Bekende verschijningsvormen van digitale identiteiten zijn OpenID en Information Card.

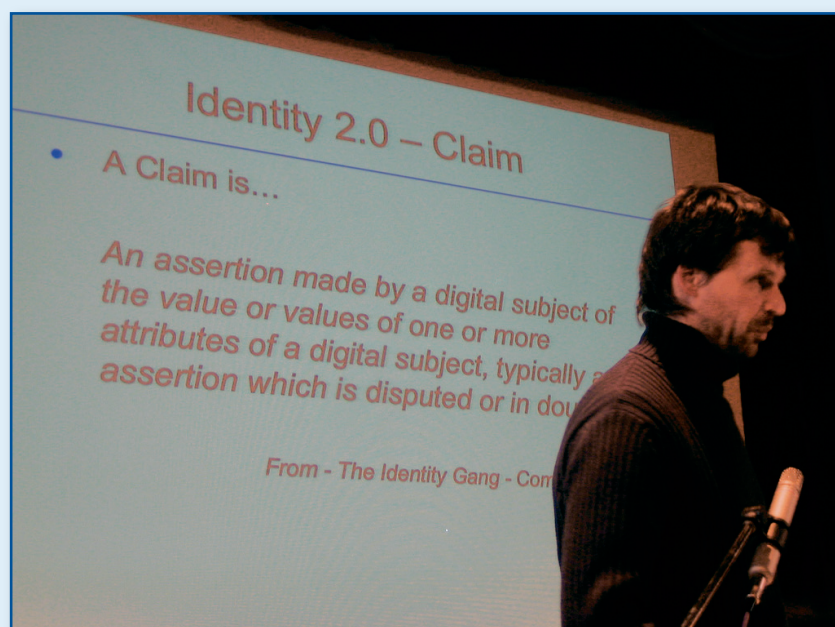
Een belangrijke standaard is SAML, Security Assertion Markup Language, een XML variant voor het uitwisselen van authenticatie- en autorisatiegegevens. Een van de berichtensoorten binnen SAML is de 'claim', een bewering over een digitale identiteit. De A in SAML staat voor Assertion, ofwel het na validering van een attribuut nemen van een beslissing, een assertion is een vaststaand feit. Een attribuut is een aan een identiteit toegekend kenmerk en heet ook wel Claim.

[3] Federation werk ik hier niet verder uit, is wellicht stof voor een andere artikel (wie schrijft?)

[4] De vraag is hierbij natuurlijk wat je een kernactiviteit vindt

Een Identity Provider verstrekt aan een individu een digitale identiteit en registreert bij die identiteit een of meer attributen, de claims. Te denken valt aan standaardinformatie als NAW of geboortedatum. De Identity Provider staat in voor de juistheid van de verstrekte informatie. Afhankelijk van het vertrouwen in de IdP is dan sprake van een bewering of van een feit. De SAML standaard is een open standaard die door alle belangrijke leveranciers die zich bezighouden met onder meer Federated Identity Management³ wordt ondersteund. Het begrip claim (bewering) is dan ook een industriestandaard en het interessante is dat we dat mechanisme nu kunnen toepassen als een drager van autorisatie-informatie.

Bron: 'Dick Hardt', Mary Hodder op Flickr.com



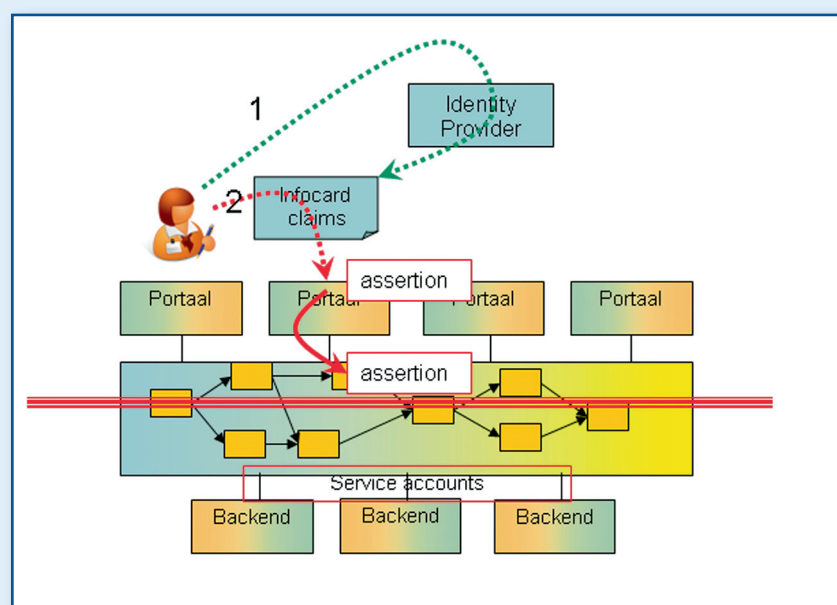
Laten we eerst even teruggrijpen naar RBAC. Een rol is feitelijk een bewering over een identiteit. Een identiteit vervult een rol in een organisatie of binnen een proces. Een rol is dus te vatten in een claim. Als een identity provider (IdP) in een digitale identiteit nu ook een rol als claim vermeldt, dan kunnen we die claim gebruiken als autorisatiemechanisme, zonder dat we in een directory op basis van een userid de security bevoegdheden hoeven uit te lezen (we hoeven alleen

de Identity Provider te vertrouwen, maar daar kom ik op terug). Daarmee hebben we dus op de website inderdaad ook geen interne directory met gebruikersinformatie (zoals Active Directory) meer nodig. Als we niet terug hoeven te grijpen naar een directory in de backend, maar kunnen vertrouwen op het identiteitenbeheer bij de Identity Provider, dan hebben we daarmee feitelijk ook bereikt dat we de identiteit zelf niet hoeven te beheren! *En daarmee vervalt meteen het eerste grote knelpunt:* we kunnen verschillende niet-beheerde identiteiten verschillende rollen geven. Mits we de door de Identity Provider verstrekte rollen (claims) kunnen herkennen. Feitelijk ga je dus een niet-kernactiviteit⁴ - het beheren van identiteiten van klanten / partners / et cetera - uitbesteden.

Vervolgens kunnen we Portaalfuncties beschikbaar stellen op basis van de rol die in het paspoort gedefinieerd is. Daarbij moet er een 'claim-autorisatie matrix' bestaan op basis waarvan de autorisaties worden geëffectueerd. Als de portaalfuncties services starten en als een gebruiker binnen het portaal alleen kan wat hij op basis van zijn claims mag en de afhankelijke services in de backendsystemen niet rechtstreeks kan benaderen, dan maakt het feitelijk

ook niet uit dat een serviceaccount alle rechten heeft. *Daarmee hebben we dus eigenlijk het tweede grote knelpunt weggenomen.* We moeten autoriseren buiten de servicebus, aan de frontends. Dus we gaan niet meer autoriseren in de backendsystemen, we gaan autoriseren in het portaal en in het proces. Dat is even wennen.

Figuur 2 Autorisatie in portaal en/of proces op basis van claims



Laten we het begrip rol nog iets verder uitdiepen. Een rol is feitelijk een verzameling autorisaties waarmee een samenhangend takenpakket kan worden uitgevoerd. Het takenpakket wordt gedefinieerd binnen de bedrijfsvoering, laten we zeggen dat een proceseigenaar aangeeft welke taken bij elkaar horen en hoe de functiescheiding is geregeld. Het maakt deze proceseigenaar eigenlijk helemaal niet uit wie de taak uitvoert. Het gaat er alleen om dat aan de kwaliteitseisen die worden gesteld aan het proces wordt voldaan. Dat is een interessante stelling. Eigenlijk hoeft de proceseigenaar helemaal niet te bepalen welke rol welke taken mag uitvoeren. Hij moet aangeven aan welke kwaliteitseisen de uitvoering moet voldoen. En dat is eigenlijk helemaal nieuw. De proceseigenaar wil eigenlijk dat iemand die een processtap uitvoert

bepaalde capaciteiten bezit, dat die persoon over bepaalde kennis en bepaalde vaardigheden beschikt. Dat de uitvoerende een bepaalde anciënniteit bezit. Oftewel: de proceseigenaar wil dat de uitvoerende bepaalde kenmerken bezit. Of dat de uitvoerende afhankelijk van bepaalde kenmerken bepaalde processtappen niet uit mag voeren of gegevens niet mag benaderen. En wat zijn dergelijke kenmerken anders dan

Claims, beweringen over een identiteit. Als de proceseigenaar nu eens niet bepaalt dat iemand een accountmanager moet zijn, maar dat een taak alleen mag worden uitgeoefend door iemand die een x-aantal jaar in dienst is en een bepaald diploma heeft, dan leidt dat meteen tot de conclusie dat het begrip rol voor de proceseigenaar geen rol speelt. Functiescheiding is wel een relevant aspect, maar met Audit Based Access Control (in combinatie met op context gebaseerde aanvullende controles zoals transactielimieten) komen we ook een heel eind als het gaat om het toewijzen van verantwoordelijkheid voor transacties. Iedereen mag alles, ik wil het alleen wel kunnen controleren.

Wat we hier effectief mee bereiken is dat we ook het kwalitatieve begrip context een plaats kunnen geven. De context

van senior of junior medewerker (jaren in dienst), of de context van juridische entiteit 1 of 2, is te vervatten in Assertions. Andere aspecten van context, zoals kanaal of soort werktijd, zijn met technische middelen wel te bepalen. *Dat betekent feitelijk dat we ook het derde en het vierde knelpunt met claims kunnen adresseren.*

We zijn er nog niet helemaal. Als je de oplossing in al zijn eenvoud beschouwt, lijkt het de ideale oplossing. Maar we zijn onze tijd te ver vooruit. Er ontbreken simpelweg enkele randvoorwaarden om dit in productie te kunnen nemen.

Ontbrekende randvoorwaarden

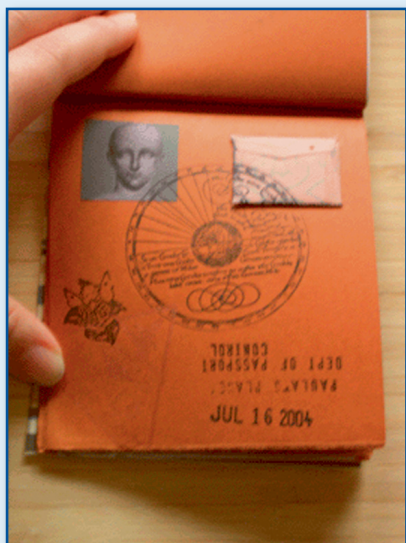
• Identity Providers

Er zijn op dit moment geen Identity Providers die digitale identiteitsbewijzen voorzien van de gewenste kwaliteit van claims (assertions) kunnen leveren. Er zijn wel Identity Providers, maar die leveren op dit moment authenticatieservices. Denk aan DigiD en EasyID van Diginotar. Dat zijn oplossingen waarbij een serviceprovider de authenticatie uitbesteedt aan de Identity Provider. Dat is niet wat we zoeken. We willen het identiteitenbeheer uitbesteden en de autorisatie in eigen hand houden. Authenticatie is in dit verhaal niet aan de orde. Begrijp me niet verkeerd: authenticatie is van belang (zie het boek Identity Crisis⁵), maar dat is een verdere verdieping van de identiteit. De authenticatiemethode kan in een claim worden vervat. We zoeken een vertrouwde partij die digitale paspoorten levert waarop wij onze autorisatie kunnen baseren. We kunnen zelf allemaal wel een Identity Provider oprichten, maar daar zijn we niet bij gebaat. Noch wij, als service provider, noch de klant. Wij raken dan het proces niet kwijt en de klant heeft nog steeds heel veel digitale identiteiten voor heel veel verschillende doeleinden. We willen graag door Identity Providers verstrekte digitale identiteiten kunnen vertrouwen. Dat betekent dan dat we de identity providers willen vertrouwen. Feitelijk zoeken we

[5] Boekbespreking Identity Crisis in Informatiebeveiliging 1 van 2009

voor Identity Providers ook iets als Cross Certification uit de PKI-wereld. Maar op dit moment bestaat een dergelijke infrastructuur niet.

Bron: 'Passport interior 2',
Jenny Factory op Flickr.com



- *Digitale portefeuilles*

Een digitaal paspoort bewaar je in een digitale portefeuille. Information Card is een prima bruikbaar paspoort, maar er zijn nog maar weinig mensen die een dergelijk paspoort kunnen gebruiken. De digitale portefeuille heet in Windows-omgevingen Windows Cardspace en die is standaard aanwezig in Windows Vista. Als dat product handiger gelanceerd was geweest, dan waren de meeste pc's nu wel voorzien van een digitale portefeuille. Helaas. Andere systemen moeten handmatig worden voorzien van de portefeuille. Windows XP heeft .NET 3.x nodig, MacOS en Linux kunnen met DigitalMe (van het Eclipse project Bandit, van Novell) uit de voeten. Maar zoals gezegd, allemaal handwerk. Dat betekent dat voor reguliere gebruikers, klanten, het digitale paspoort in de vorm van Information Card misschien wel een brug te ver is. Een alternatief is OpenID, maar dat is ook een authenticatieservice en daar ben ik geen

voorstander van, een dergelijke service kent immers je gedrag op internet.

- *Audittrail*

Er zijn nog geen standaarden voor de audittrail. Dit instrument is van belang om elke transactie in een backofficesysteem te kunnen terugvoeren op een uniek uitgevoerd proces en binnen dat proces op alle binnen dat proces acterende identiteiten. Het inrichten van een audittrail is ook weer een verhaal op zich. Je zult bij voorbeeld vast moeten leggen:

- Datum/tijd
- Gebruikersnaam (van de portaalgebruiker)
- Procesidentificatie (unieke identificatie van de 'instance' van een gestart proces)
- Transactie-identificatie (unieke identificatie van de transactie zoals die in het backendsysteem ontstaat of wordt beheerd)

Op die manier kun je via de audittrail in de SOA altijd vaststellen wie binnen welk proces heeft geacteerd. En wie een hand heeft gehad in de transactie die in het backendsysteem is vastgelegd.

- *Standaarden voor claims*

Er zijn nog geen uitgewerkte standaarden voor claims. Information Card van Microsoft kent wel diverse standaard claims, maar die zijn niet voldoende om als basis voor autorisatie te dienen. Standaardclaims zullen ook niet zomaar toegepast kunnen worden, omdat autorisaties natuurlijk procesgeoriënteerd zijn. Standaardclaims kunnen alleen aan gestandaardiseerde autorisaties worden gekoppeld. De vraag is of het mogelijk zal zijn om te komen tot een min of meer beperkte standaardisering. Voor interne verstrekte claims door interne Identity Providers is dat natuurlijk geen probleem. Niets weerhoudt een organisatie ervan om digitale paspoorten met een claim 'Medewerker = accountmanager' te verstrekken.

Conclusie

Claims Based Access control maakt het mogelijk om binnen een SOA/Web 2.0 omgeving toegangsbeveiliging op een effectieve manier te realiseren.

De technieken zijn beschikbaar, de standaarden zijn aanwezig. Het vergt wel enkele onconventionele methoden en de vraag is wanneer deze methoden in praktijk gebracht zullen worden. We kunnen wel willen, maar of we willen kunnen, is de vraag.

Voor de korte termijn levert dat ook wel wat discussie op. Mag het wel? Ofwel: zijn we eraan toe, accepteren stakeholders en toezichhouders dit samenspel van beheersingsmaatregelen?

Als we voorbijgaan aan het knelpunt van standaardisering dat ontstaat bij het ontsluiten van een SOA voor Web2.0 gebruikers op het internet, dan kunnen we het mechanisme wel al inzetten voor autorisatie binnen een SOA. Daarmee kunnen we diverse knelpunten alvast wegnemen. Ik zou zeggen: denk hier eens over na en aarzel niet om te reageren.