

ABAC, meer ideetjes

André Koot > André Koot is Security manager bij Univé Verzekeringen in Zwolle.

In dit artikel ga ik in op het in het vorige nummer van Informatiebeveiliging verschenen artikel over 'Audit Based Access Control in de zorg' van Marnix Dekker, Thijs Veugen en Sandro Etalle. Ik zou met u mijn gedachten willen delen over de vraag of ABAC niet voor meer doelen bruikbaar is dan waarvoor ABAC in het vorige artikel werd beschreven? Voor mij is dat eigenlijk geen vraag, maar dan moet ik even terug in de tijd (niet zo heel ver overigens) om uit te leggen hoe we binnen Univé omgaan met de nieuwe werkwijzen die we toepassen bij de ontwikkeling van informatiesystemen. En daarbij raken we onder meer de opmerking van de auteurs dat ABAC niet bruikbaar is voor web-services op het internet. Misschien...

Univé Informatiebeleid

Vanuit een nieuwe visie op de positionering van Univé als totaalaanbieder van verzekeringsproducten is gewerkt aan het ontwikkelen van een architectuur die deze nieuwe visie realiseerbaar maakt. Die architectuur is gevonden in de Service Oriented Architecture, SOA. Omdat wij ook zorgverzekeraar zijn, kunnen wij die afkorting wel toepassen.



Waar het kort gezegd op neerkomt, is dat we informatiesystemen hebben opgesplitst in verschillende lagen. Zo is omwille van het multidistributie principe gekozen voor een opdeling van de presentatiefunctie in diverse front-ends voor de diverse in- en externe doelgroepen. Vanuit de wens om Operational Excellence na te streven is een proceslaag ontwikkeld, waardoor we bijvoorbeeld een eenmaal ontwikkeld werkproces voor verschillende kanalen en processen in kunnen zetten. De onderste laag is de Bedrijfsfunctie laag. Ook deze functies worden zo ontwikkeld dat herbruik-

baarheid gegarandeerd is en dat het mogelijk is om een versie van een bedrijfsfunctie te vervangen door een andere versie, terwijl de front-ends en de proceslaag ongemoeid blijven. Dat blijkt nuttig gezien bijvoorbeeld de migratie van ons Zorgsysteem naar het Oracle OpenZorg systeem dat het nieuwe strategische platform wordt voor onze zorgverzekeringen.

Deze onderste laag wordt ontsloten via webservices. Daarbij hebben we gekozen voor het dotNet platform van Microsoft.

We hebben in anderhalf jaar heel veel ervaring opgebouwd en een groot aantal bedrijfsprocessen wordt inmiddels via het SOA platform afgehandeld.

Er is echter een 'maar'. In het Informatiebeleidsplan dat ten grondslag ligt aan de SOA architectuur is Informatiebeveiliging prominent aanwezig als randvoorwaarde. Kernbegrippen daarbij zijn Autorisatie en Rollen. Dat tendeeft naar Role Based Access Control en dat fenomeen wordt bij ons dan ook onderzocht op toepasbaarheid.

SOA en RBAC

In een SOA ontstaat net zoals bij Multi Tier systemen de situatie dat (als het goed is) een gebruiker niet zelf toegang tot een database verkrijgt, maar dat een (systeem)proces de gebruiker is van een database. Dat systeemproces moet dus alle benodigde auto-

risaties hebben om alle transacties binnen een informatiesysteem te kunnen uitvoeren. Die transacties worden door een systeemproces uitgevoerd, hetgeen in de toegangslog van de database ook als zodanig is terug te vinden. Vanuit de optiek van beveiliging van de database is dat een prima oplossing, er hoeven immers geen individuele rechten te worden beheerd, een gewone gebruiker mag niets binnen de database. Dat zal elke database beheerder aanspreken.

Dit levert wel een probleem op, want als het systeemproces alle autorisaties heeft, dan heeft de gebruiker die het proces start dus feitelijk ook alle autorisaties om de gegevens te benaderen. Nee, daar moeten we dan ook ingrijpen om problemen te voorkomen.

Om dat probleem op te lossen zijn verschillende oplossingsrichtingen denkbaar. Een gebruiker zou alleen die functies mogen kiezen waarvoor hij geautoriseerd is, of de autorisatie van een gebruiker wordt gecontroleerd na het kiezen van een functie, maar voor het uitvoeren van een transactie. Dat laatste is ergonomisch niet heel fraai, dus vandaar dat het opbouwen van een gebruikersinterface waarin de toegestane functies worden gepresenteerd, de voorkeur heeft.

In de traditionele client server systemen is dat betrekkelijk eenvoudig, omdat de autorisatiestructuur binnen de gebruikerssessie beschikbaar is. De applicatie (thin of fat client) weet op voorhand welke autorisaties beschikbaar zijn en presenteert uitslui-

tend die functies die bij de autorisatie van de gebruiker horen.

In het geval van een SOA is er geen persistentie van de autorisatiegegevens. Daarvoor zijn verschillende redenen. Ten eerste wordt gebruikgemaakt van webtechnologie. De applicatie is niet meer dan een webpagina die in een browser wordt gepresenteerd. Er worden wel enige sessiegegevens vastgehouden (bijvoorbeeld in een cookie), maar de autorisatiestructuur hoort daar niet bij.



Ten tweede is er geen vaste applicatie. Door het gebruik van webservices ontstaat er een mogelijk onvoorspelbaar portfolio van bedrijfsfuncties en services waarvan feitelijk niet bekend is hoe die in elkaar zitten. De interface van zo'n service moet natuurlijk wel bekend zijn (je moet die service immers kunnen aanroepen en het resultaat ervan moet je kunnen interpreteren), maar de logica van de achterliggende applicatie of het achterliggende datamodel is misschien niet eens bekend. Voorbeeld: we maken voor offertes voor personenautoverzekeringen gebruik van een externe webservice die de catalogusprijs van een auto bepaalt aan de hand van het kenteken van een auto. Die gegevens beheren wij niet, die hele service komt van buiten. We hebben geen zicht op de structuur en logica van die applicatie.

Het probleem van de autorisatiefunctie binnen SOA

Conclusie van dit fraais: een autorisatiefunctie zal er anders uit moeten zien dan in de traditionele omgeving. Sterker: een autorisatiefunctie in een SOA zal zelf ook een service moeten zijn, maar dan wel een binnen het beveiligingsdomein vertrouwde service, die op basis van een autorisatieschema kan bepalen of iemand de

transactie mag uitvoeren. Het moet wel een service zijn, want we weten op voorhand niet op welke wijze iemand een proces binnenkomt, maar we weten wel dat een 'service bus'-functie toegang tot de services biedt. De Service Bus doet dat door services aan te spreken en dat is dan ook de manier waarop we de autorisatiefunctie moeten implementeren.

De autorisatiefunctie is dus een webservice. Maar die webservice heeft geen kennis van de persoon die de transactie start. Hoe kun je een transactie toestaan als je geen kennis hebt van het individu die geautoriseerd moet worden?

Dat is dan meteen ook de achtergrond voor de opmerking van de auteurs van het ABAC artikel over de ontoreikendheid van ABAC voor webservices. En dat klopt dan natuurlijk ook: hoe kun je transactiegegevens registreren als je niet weet wie het individu is die de transactie initieert?

Dat probleem is in een gecontroleerde SOA omgeving echter wel op te lossen, mits de identiteit van het individu op de een of andere manier kenbaar wordt gemaakt en mits de authenticiteit van die identiteit gewaarborgd is. Gelukkig is dit probleem te ondervangen door in het bericht dat door de service bus wordt getransporteerd de identificatie mee te sturen. Dat is door toepassing van het XML protocol natuurlijk geen probleem. Resteert het probleem van de authenticiteit zelf. Als de hele SOA en de gebruikers ervan binnen één en hetzelfde beveiligingsdomein opereren (bijvoorbeeld binnen de Active Directory), dan hoeven we alleen maar te vertrouwen op de aanmelding op de werkplek en dat trucje kennen we al jaren. Niet dat dat gewaarborgd is, maar het niveau van beveiliging van een SOA wijkt op dit punt in het geheel niet af van de traditionele infrastructuur. Als we de identiteit binnen een gewoon systeem als Siebel of Oracle vertrouwen, dan kan dat ook in een SOA...

Omdat de autorisatiefunctie een centrale rol vervult, moet het ook een vertrouwde service zijn. Met name dit aspect leidt ertoe dat er een vertrouwensrelatie tussen de service bus en de achterliggende webservices moet bestaan. Bij ons betekent dat bijvoor-

beeld dat er een PKI infrastructuur wordt ingericht om dat vertrouwen te leveren. Maar dat terzijde.

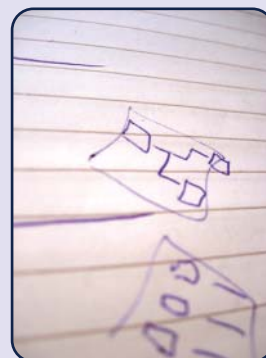
Waarom ABAC in plaats van RBAC?

In het Informatiebeleid wordt uitdrukkelijk de keuze gemaakt voor rolgebaseerd autoriseren. Dit is onder meer terug te voeren op mainframe systemen waarin feitelijk iedere medewerker op individuele basis wordt geautoriseerd voor het uitvoeren van een taken. Dat werkt op zich prima, maar het beheer is te complex. Het tussenschakelen van het niveau 'rol' is dan ook een logische gedachte.

Hoe de rol te beschouwen?

Uitgaande van een groene weide theorie ontwerpt een proceseigenaar een bedrijfsproces en hij zal daarin de hem bekende randvoorwaarden meenemen. Daaronder zijn eisen vanuit onder meer controletechnische functiescheiding te vatten. De proceseigenaar is verantwoordelijk voor het goede verloop van een productieproces, bijvoorbeeld een offerteproces. Daarbij worden processtappen en rollen die die processtappen uitvoeren onderkend. Het proces wordt vervolgens gemodelleerd in ons workflowmanagement product. Daarbij worden geautomatiseerde en handmatige processtappen ingebouwd. Het proces wordt gestart vanuit een front-end systeem en binnen de proceslaag van de SOA uitgevoerd. Het resultaat van de workflow stappen wordt bijvoorbeeld in werkbakjes in een front-end gepresenteerd.

In dit model definieert de proceseigenaar de rollen. Laten we het referentierollen noemen. Als de uitvoeringsorganisatie vervolgens deze referentierollen toekent aan medewerkers, dan is de autorisatie op grond van het RBAC principe een feit.

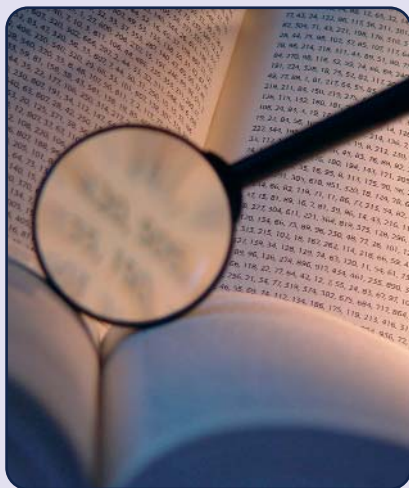


Een complicatie ontstaat als de uitvoeringsorganisatie rollen toekent aan medewerkers, zonder het referentiemodel toe te passen. Dat is een zeer plausibele situatie: binnen de Univé groep zijn er 34 verschillende uitvoeringsorganisaties, die ieder organisatorisch van elkaar af kunnen wijken. Elke directeur van een uitvoeringsorganisatie (bijvoorbeeld een lokale Onderlinge Brand WaarborgMaatschappij) is zelf verantwoordelijk voor zijn eigen beheer en legt verantwoording af aan diverse toezichthoudende partijen, zoals de Nederlandsche Bank. Een proceseigenaar heeft hiërarchisch geen enkele zeggenschap op de uitvoering door uitvoeringsorganisaties. Het kan dan ook voorkomen dat de feitelijke roltoekenning afwijkt van het referentiemodel.

In ons geval weten we al dat er afwijkingen bestaan. Maar als een proces ook door derden zou kunnen worden uitgevoerd, dan weten we feitelijk niet eens of de feitelijke inrichting afwijkt van het referentiemodel. Wellicht past het referentiemodel niet eens in die uitvoeringsorganisatie. Er bestaan dan twee mogelijkheden: ofwel we dwingen toepassing van het referentiemodel af of we accepteren het feit dat de uitvoering door een derde partij strijdig kan zijn met de referentie. Dat impliceert wel dat we de eigen verantwoordelijkheid van die derde als uitgangspunt hanteren. En dat betekent dan weer dat we feitelijk alleen een theoretische referentie RBAC kunnen ontwikkelen. En dat betekent dat een proceseigenaar op geen enkele manier kan weten of zijn proces op de juiste manier wordt uitgevoerd. RBAC is dus niet de oplossing gezien vanuit de proceseigenaar.

We hebben de volgende gedachtegang gevolgd:

Als een proceseigenaar de rolscheiding niet kan afdwingen, dan moet hij feitelijk accepteren dat iedereen alles kan. Maar om zijn verantwoordelijkheid te kunnen dragen zal hij toch toezicht moeten houden. Dat betekent dat er een vorm van logging moet worden bijgehouden waarmee hij achteraf zelf toepassing van zijn 'RBAC'-regels kan toetsen. Wellicht kan er zelfs een real-time alerting mechanisme worden gebruikt, zodat hij meteen op de



hoogte is van het doorbreken van bijvoorbeeld de controletechnische functiescheiding. Hoe hij vervolgens daarmee omgaat, is een andere vraag. En wellicht kan een dergelijke 'inspectie' functie ook als dienst worden aangeboden aan de uitvoeringsorganisatie, waardoor ook die organisatie kan toetsen of voldaan wordt aan de interne regels.

Een dergelijke werkwijze is overigens niet nieuw. Feitelijk hanteren we in het algemeen dezelfde manier als de controle op het internetgebruik. Werd in het verleden het gebruik alleen toegestaan aan aangewezen personen, nu mag iedereen het en kijken we alleen wat er niet voldoet aan de normen.

Er is een tweede vergelijking: een Intrusion Detection en Prevention systeem doet ook iets soortgelijks: we definiëren een beleid en een slim systeem controleert welk netwerkverkeer niet voldoet aan het beleid. Dat leidt dan tot een event of alert. Dat mechanisme lijkt dus wel wat op ons beoogde controlesysteem: we definiëren een beleid van referentierollen en we controleren achteraf (of real-time) of het beleid doorkruist wordt.

Randvoorwaarden

Wil dit systeem werken, dan moet er wel iets worden gedaan aan een fundamenteel probleem, dat ook in het ABAC artikel wordt geadresseerd. Het systeem staat of valt met de betrouwbaarheid van de identiteit die in een proces wordt ingeschakeld. Identificatie en authenticatie zijn van vitaal belang. Binnen een domein-infrastructuur is dat, zoals ik al eerder schreef, niet zo'n groot probleem. We hanteren hetzelfde niveau van vertrou-

wen als dat nu wordt geaccepteerd in de traditionele omgevingen. Dat doen we ook in de webservice architectuur.

Buiten een dergelijke omgeving is er wel een probleem. Hoe weten we wie in een proces iets doet als we de identiteit niet kennen? Bijvoorbeeld als het proces wordt gestuurd vanaf het internet.

Je kunt denken aan een inlogschermje (maar ja, wat levert dat aan vertrouwen op?) of aan het gebruik van certificaten (maar ja, wie wil dat beheren?).

Laat ik maar inzetten op toepassing van User Centric Identity Management (dit wordt beschreven in het artikel over dit onderwerp vanaf bladzijde 7 in dit nummer van Informatiebeveiliging), waarbij voor dergelijke productieprocessen vereist wordt dat een Infocard van een managed Identity Provider wordt gebruikt.

Conclusie

De ideeën in dit artikel zijn beslist niet volledig uitgewerkt. Wel is al onderzocht of en zo ja met welke hulpmiddelen bijvoorbeeld de loganalyse kan plaatsvinden. Er zijn al producten op de markt die diensten kunnen bewijzen. Te denken valt aan een open source pakket als Splunk, of commerciële oplossingen als Cisco Application Oriented Networking of Cognos. Deze producten zijn niet specifiek voor ABAC ontwikkeld, maar wel bruikbaar en beschikbaar. Dat maakt het interessant.

Ook moeten we uitzoeken of het principe dat iedereen alles kan, maar niet zou mogen, bruikbaar is. Ongeacht de te kiezen methode, zul je toch moeten aantonen 'in control' te zijn.

We zijn er bij Univé in ieder geval nog niet uit, maar we vermoeden dat een ABAC methode in een SOA omgeving beter te implementeren is dan een RBAC oplossing. Laten we elkaar op de hoogte houden van de vorderingen op dit gebied.