Architecten, netwerkbeheerders, CIO's en andere betrokkenen hebben hun mond vol van PKI, firewalls, SSL en S/MIME om maar een paar aan beveiliging gerelateerde trends te noemen. Voldoen deze oplossingen dan niet voor een services architectuur? Wat maakt dat de standaardstrategieën voor beveiliging in bepaalde gevallen niet toereikend zijn? In dit artikel gaan we nader in op de specifieke uitdagingen die een servicesarchitectuur stelt aan de beveiliging van servicegeoriënteerde systemen.

Spanningsveld tussen flexibiliteit en veiligheid

Uitdagingen bij de beveiliging van een services-architectuur

In ons vorige artikel over servicesarchitectuur (zie Software Release Magazine nr. 4/2005) hebben we beveiliging benoemd als één van de grootste hindernissen op weg naar de implementatie van een services architectuur. Het spreekt voor zich dat een open architectuur als een servicesarchitectuur moeilijker te beveiligen is dan een meer gesloten architectuur. Niettemin heeft beveiliging de laatste jaren volop in de aandacht gestaan, en zijn er technisch ook behoorlijke vorderingen gemaakt.

SPECIFIEKE UITDAGING Er is een aantal aspecten te onderscheiden die maken dat een services architectuur lastiger te beveiligen is dan een meer traditionele, op componenten gebaseerde of objectgeoriënteerde architectuur.

In de eerste plaats praten we vaak over services, waarvan bij het ontwerp en de ontwikkeling nog niet (volledig) bekend was waar en hoe ze gebruikt zouden worden. De consumers van de service stonden nog niet vast, net zo min als de volledige context waarbinnen de service gebruikt zou gaan worden. Daarnaast worden services vaak ingezet in heterogene omgevingen, waar verschillende authenticatie- en autorisatiemechanismen gebruikt worden, die vaak onderling niet of niet goed kunnen samenwerken, in verschillende talen zijn geschreven of verschillende interfaces hebben. In de derde plaats zijn deze bestaande authenticatie- en autorisatiemechanismen niet meer bruikbaar vanwege de extra abstractielaag die de servicesarchitectuur kent, in de vorm van de zogenaamde *services layer*. Deze abstractielaag, die onder meer verantwoordelijk is voor de compositie van services, abstraheert de onderliggende complexiteit van de technologische implementatie, waardoor services op een gestandaardiseerde manier met elkaar kunnen communiceren en aangeroepen kunnen worden. De keerzijde is dat ook de *security context* van de onderliggende applicaties geabstraheerd wordt: de bestaande manieren van applicaties om aanroepen en aanroepers te authentiseren en vervolgens te autoriseren, komt daarmee in één klap te vervallen.

In de vierde plaats maken servicegeoriënteerde omgevingen bij voorkeur gebruik van open, webgebaseerde protocollen om de services met elkaar te laten communiceren. HTTP was lange tijd het favoriete protocol, SMTP en FTP zijn ook zeer wel mogelijk, en de laatste tijd is TCP als protocol voor SOAP-berichten sterk in opkomst, mede vanwege de superieure performance. Over de beveiligingsrisico's van deze protocollen, zeker als ze ook nog eens worden ingezet via of gekoppeld aan het 'open' internet, kan een beveiligingsdeskundige boeken volschrijven.

In de laatste plaats is een servicegeoriënteerde omgeving sterk gericht op koppeling met externe partijen in een keten, waardoor er sterke afhankelijkheden bestaan van de beveiligingsniveaus van de ketenpartners. Iedere keten heeft uiteraard een zwakste schakel, waardoor de mate van beveiliging van de gehele keten bepaald wordt door de mate waarin deze zwakste schakel beveiligingstechnisch goed in elkaar zit.

Uit het bovenstaande wordt duidelijk dat beveiliging van een servicegeoriënteerde omgeving nieuwe uitdagingen met zich meebrengt, en dat oplossingen voor een meer traditionele, op componenten gebaseerde architectuur, niet meer (volledig) toereikend zijn.

CONTEXT VAN SOA-BEVEILIGING In ons vorige artikel hebben we gezien dat vraagstukken ten aanzien van beveiliging van services kunnen teruggebracht worden tot een beschouwing van de volgende vijf blokken:

1. *Onweerlegbaarheid* (in vakjargon ook wel non-repudiation genoemd) heeft betrekking op maatregelen die borgen dat een entiteit niet kan ontkennen dat het een bericht heeft ontvangen of heeft verstuurd.
2. *Vertrouwelijkheid* moet garanderen dat niet-geautoriseerde entiteiten berichten niet kunnen lezen.
3. *Autorisatie* bepaalt welke privileges beschikbaar zijn voor een geauthentiseerde entiteit.
4. *Authenticatie* garandeert dat een entiteit (een persoon of een systeem) daadwerkelijk is wie of wat hij beweert te zijn.
5. *Integriteit* garandeert dat documenten of berichten niet gewijzigd zijn.

Bij elk project of combinatie van systemen zal een andere selectie uit de bovenstaande lijst een meer of minder belangrijke rol spelen. Alles draait bij de beveiliging om risico's vermijden. Echter hoe meer risico's vermeden moeten worden, hoe meer in beveiliging zal moeten worden geïnvesteerd. Niet alleen zullen er financiële afwegingen moeten worden gemaakt, ook staat beveiliging immers vaak op gespannen voet met bruikbaarheid. Op dit vlak zal een balans moeten worden gevonden.

Daarnaast is er de intrigerende en steeds weer terugkerende kwestie van de losse koppeling ofwel *loose coupling*. Hoe meer specifieke beveiligingskennis en -informatie uitgewisseld en gedeeld wordt door de deelnemende partijen en services binnen een servicesarchitectuur, hoe meer dat ten koste lijkt te gaan van deze losse koppeling. In eerdere artikelen hebben we gezien dat de mate van koppeling afhangt van de hoeveelheid gedeelde kennis die vereist is voor services om onderling te communiceren. Zodra er veel beveiligingsgerelateerde informatie gedeeld wordt (zoals *tokens*, sleutels of certificaten), gaat dit onherroepelijk ten koste van de losse koppeling die wordt nagestreefd met een services architectuur.

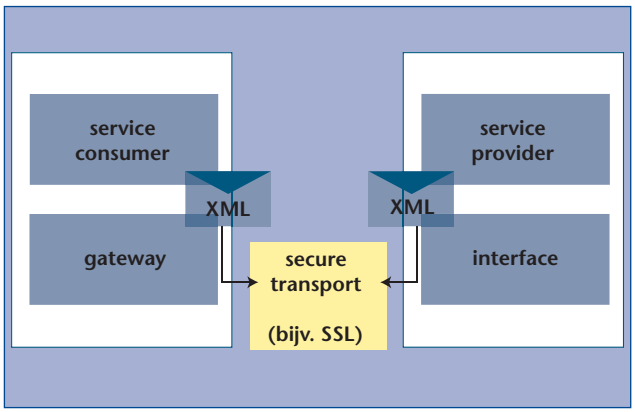
Laten we vanuit de geschetste problematiek en de vijf beschouwinggebieden eens kijken naar de verschillende oplossingsrichtingen die er zijn om een goede mate van beveiliging te realiseren.

BEVEILIGINGSSTRATEGIEËN BINNEN EEN SOA

Webservices staan voor veel organisaties centraal in de plannen rondom adoptie van een servicesarchitectuur. Hoewel een servicesarchitectuur ook zonder een enkele webservice gerealiseerd kan worden, gaan we voor het gemak hier uit van realisatie van een servicesarchitectuur via toepassing van webservices. Daarnaast zijn de strategieën voor het beveiligen van webservices breder toepasbaar dan alleen voor webservices, omdat vrijwel alle standaarden rondom webservice-beveiliging open zijn, of op de nominatie staan om verheven te worden tot open standaard. Dé manier in onze optiek om een servicesarchitectuur te beveiligen, is dan ook via open (webservice) standaarden.

Om webservices te beveiligen kunnen we drie strategieën onderscheiden: het beveiligen van het transport, het beveiligen op applicatieniveau, en het beveiligen van het bericht zelf.

TRANSPORT LEVEL-BEVEILIGING Bij deze manier van beveiliging wordt het beveiligingsmechanisme van het transportmedium zelf gebruikt. We kunnen hierbij denken aan bijvoorbeeld toepassing van SSL of IP/SEC.



FIGUUR 1. Transport level-beveiliging.

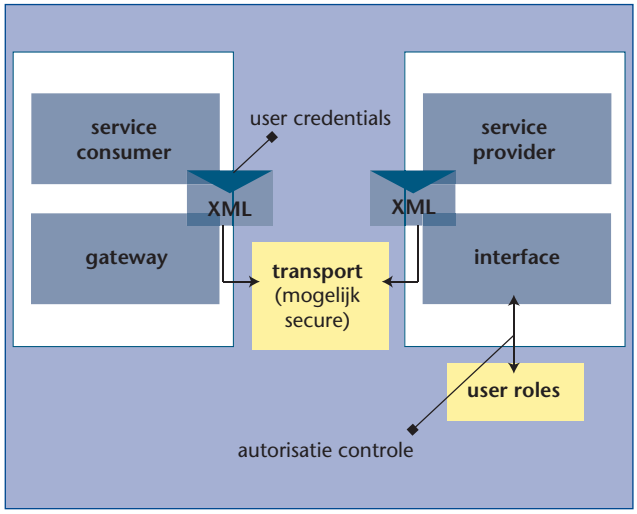
Deze manier van beveiligen is aantrekkelijk vanwege de relatieve eenvoud ervan: de transportlaag draagt min of meer zorg voor de beveiliging van de uitwisseling. Hoewel voor de eenvoud veel te zeggen is kleven er toch wat nadelen aan deze manier.

Hoewel SSL kan worden gebruikt voor eisen als authenticatie, integriteit en vertrouwelijkheid, is SSL alleen geschikt voor een zogenaamde *point-to-point* beveiliging. Zodra het bericht is afgeleverd bij de bestemming of het eindpunt van de verbinding, is het bericht niet meer versleuteld en beveiligd. Het bericht is alleen versleuteld voor zolang als de sessie tussen zender en ontvanger duurt.

Bovendien is er buiten deze *point-to-point* verbinding geen beveiliging waar de eigen organisatie controle op heeft. Dit lijkt afdoende voor een intern gerichte service-georiënteerde architectuur, maar niet voor een meer extern gerichte, waar wellicht meerdere tussenstations (intermediairs ofwel *service brokers* bijvoorbeeld) nodig zijn. Binnen SSL kunnen tevens de berichten alleen in hun geheel worden versleuteld, hetgeen een nadeel kan zijn. Bijvoorbeeld bij routering op basis van inhoud van het bericht (zogenaamde *content-based routing* of *itinerary-based routing*) moet immers dan het hele bericht worden ontcijferd om te zien waar het bericht heen moet.

BEVEILIGING OP APPLICATIENIVEAU De tweede methode betreft het delen van identiteiten tussen twee systemen die met elkaar communiceren via webservices. Binnen deze methode wordt de *security context* van de consumer meegegeven worden bij de aanroep van de provider, bijvoorbeeld via SOAP-headers. Omdat deze services vallen onder dezelfde *security provider* (bijvoorbeeld de organisatiebrede directory server), kan de provider de *security context* associëren met een gebruiker en met bijbehorende rechten.

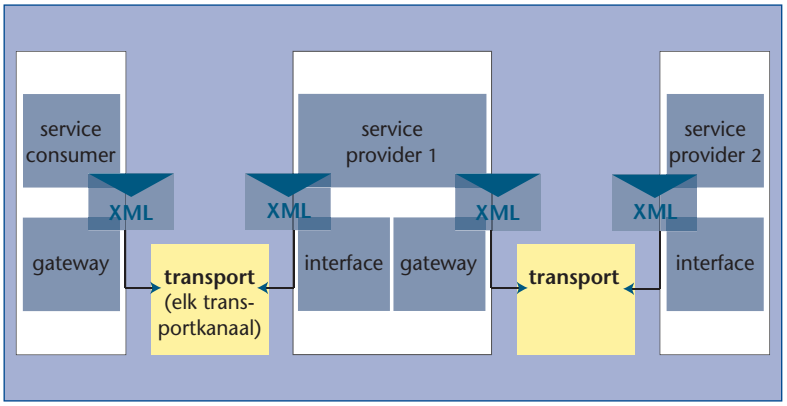
Eerder in dit artikel hebben we echter gezien dat het beveiligen van services op applicatieniveau breekt met een aantal richtlijnen voor services binnen een servicegeoriënteerde omgeving. De mate van koppeling kan moeilijk los genoemd worden, en de technologische details van de onderliggende applicaties worden niet geabstraheerd. Bovendien vereist deze aanpak dat consumer en provider dezelfde *security provider* hebben. Kortom: deze aanpak hangt van afhankelijkheden aan elkaar, en is daarom minder geschikt voor een servicegeoriënteerde omgeving.



FIGUUR 2. Beveiliging op applicatieniveau.

BERICHTBEVEILIGING Bij de methode van berichten beveiliging (zogenaamde *message level security*) wordt de beveiliging in het bericht zelf geïntegreerd. Dit wordt mogelijk gemaakt door de zogenaamde *SOAP message enhancements* (in goed Nederlands: uitbreidingen). Deze *enhancements* stellen ons in staat het bericht op te delen in diverse delen welke gerelateerd kunnen worden aan diverse beveiligingsmaatregelen. Hierdoor kan een willekeurige combinatie worden gebruikt van drie technieken:

- Meesturen van een (of meerdere) *security token(s)* voor authenticatie
- Digitaal ondertekenen *(signing)* van (delen) van berichten
- Versleutelen van (delen van) berichten



FIGUUR 3. Berichtbeveiliging.

Het voordeel van het in delen kunnen *signen* of encrypten van berichten is dat bij services waarbij meerdere partners een rol spelen elke partner zijn eigen verantwoordelijkheid kan nemen in zijn deel van het totale bericht. Uiteraard zijn deze delen vooraf gespecificeerd.

Zo kan bijvoorbeeld een service die verantwoordelijk is voor de routering van berichten, alleen het voor hem bestemde deel van het bericht ontcijferen, lezen waar het bericht heen moet, en het vervolgens doorsturen naar de betreffende bestemming. De daadwerkelijke inhoud van het bericht, kan dan uitsluitend worden ontcijferd en verwerkt door de ontvangende service. Een vergelijkbare constructie kunnen we ons voorstellen in een situatie waarbij er een zogenaamde *service aggregator* is ingezet, die zich richting consumer als één service aanbiedt, maar onderliggend zelf gebruik maakt van meerdere services.

SPECIFICATIES Voor al deze drie technieken zijn standaarden en specificaties gedefinieerd, die uitleggen hoe de genoemde technieken in de berichten kunnen worden geïmplementeerd.

Voor het gebruik van *tokens* kan bijvoorbeeld gebruik gemaakt worden van de OASIS-standaard voor de X.509 Certificate Token Profile. Hiermee kunnen berichten worden geauthentiseerd. Voor de encryptie kan gebruik gemaakt worden van de XML Encryptie-specificatie van de W3C. De versleutelde gegevens komen binnen XML-tags te staan die gedefinieerd zijn in de genoemde specificatie. Door middel van de encryptie kan de vertrouwelijkheid van een (deel van een) bericht worden gegarandeerd.

Voor het digitaal ondertekenen *(signen)* kan worden verwezen naar de XML Signature-specificatie van dezelfde W3C. Met behulp van deze techniek kan de integriteit van een (deel van een) bericht worden gewaarborgd. Tevens kan de oorsprong van het (deel van het) bericht worden vastgesteld, waarmee we ons ook kunnen wapenen tegen weerlegbaarheid.

De *WS-Security* specificatie, welke is gedefinieerd door OASIS-groep combineert een aantal aspecten in zich. *WS-Security* maakt gebruik van zowel XML Encryptie als XML Signature en levert tevens een mechanisme om security tokens mee te geven aan berichten. Hiermee kunnen dus zowel vertrouwelijkheid, integriteit als authenticatie worden ondersteund.

Tot slot noemen we nog de *SAML* (Security Assertion Markup Language), welke ook is gedefinieerd door de OASIS Groep. Deze standaard specificeert de manier waarop een *Single Sign On* (SSO) kan worden gerealiseerd. Autorisatie en authenticatie kunnen hiermee worden ondersteund.

De manier waarop beschreven kan worden wat de afspraken zijn om berichten te beveiligen volgt ook weer in een standaard: de *WS-Policy*. Deze standaard is mede opgesteld door de grootmachten IBM, Microsoft, VeriSign en RSA Security samen en biedt een raamwerk voor het definiëren van Web Service beveiligingsafspraken (zogenaamde *policy's*).

Noemswaardig in deze context zijn nog de *WS-Trust* en *WS-SecureConversation* specificaties. De *WS-Trust* specificatie heeft tot doel te ondersteunen dat systemen op een eenvoudige manier tot een betrouwbare uitwisseling van berichten komen met behulp van het gebruik van *security tokens*.

De *WS-SecureConversation* specificatie beoogt het kunnen definiëren van een *security context* waarbinnen berichten kunnen worden uitgewisseld. Langs deze weg wordt gedurende een sessie (conversatie) eenmalig over en weer beveiligingsgegevens uitgewisseld (ook wel *handshake* genoemd), waarna de uitwisseling van de 'echte' gegevens kan plaatsvinden. *WS-SecureConversation* is qua werking vergelijkbaar met SSL, echter het verschil is dat *WS-SecureConversation* onafhankelijk is van het onderliggende protocol, lees HTTP: het werkt ook met andere protocollen als TCP en SMTP.

Specificatie	Steller	Ondersteund door	Beveiligingsaspect
XML Encryption	W3C	SS4J Apache-XML-Security-J Baltimore Key tools VeriSign	Vetrouwelijkheid
XML Signature	W3C	XSS4J Apache-XML-Security-J Baltimore Key tools VeriSign	Integriteit
WS-Security	OASIS	Apache WSS4J (Axis) Trust Service Integration Kit (TSIK)(VeriSign) Weblogic Websphere Microsoft	Integriteit Vertrouwelijkheid Authenticatie
SAML	OASIS	OpenSAML	Authenticatie Autorisatie

TABEL 1. Een overzicht van de genoemde specificaties

HOBBELS Een mogelijke hobbel bij de implementatie van de berichtenbeveiliging is de nadelige invloed op de performance. Voor het heen en weer uitgeven van keys en het versleutelen van berichten is wel de nodige processorcapaciteit vereist. Hier geldt dat een balans moet worden gevonden tussen functionaliteit (in de vorm van performance) en veiligheid.

Daarnaast valt op dat er erg veel standaarden in ontwikkeling zijn, deze standaarden niet allemaal even ontwikkelaarvriendelijk zijn, en dat sommigen zelfs door de bomen het bos niet meer zien. Ook hier halen we weer ons vaste advies van stal voor wat betreft webservice-standaarden:

- Code-generatoren en toolkits ontwikkelen zich razendsnel, en bevatten meestal vrij vroeg de meest bruikbare en meest toegepaste specificaties.
- De meeste webservices hebben slechts een fractie van de beschikbare specificaties nodig. Wees kritisch over welke specificaties wel, en welke niet gevolgd worden! Zeker rondom beveiliging is er veel overlap in wat de standaarden beogen te doen op het gebied van beveiliging.

SAMENVATTING EN CONCLUSIE Samengevat zijn de specificaties en oplossingen zoals weergegeven in tabel 1 in onze optiek significant bij het beveiligen van berichten in een servicesarchitectuur. Dit overzicht is zeker niet uitputtend, en deze maatregelen alleen bieden *niet* voldoende beveiliging. Additionele maatregelen zijn nodig, maar om de specifieke kenmerken van servicegeoriënteerde beveiliging optimaal te ondersteunen, zijn oplossingen op het gebied van berichtbeveiliging cruciaal.

Omdat de beveiliging in het bericht zelf zit ingebakken, worden we onafhankelijk van het transportmedium, waardoor een heterogeen opgebouwd systeem kan worden beveiligd. Een *end-to-end* oplossing is ook mogelijk, mede doordat verschillende delen van een bericht kunnen worden beveiligd en ook weer omdat de beveiliging in het bericht zelf is gedefinieerd. De verschillende encryptie- en *signing*-mogelijkheden bevorderen het gevoel van integriteit en vertrouwelijkheid en bieden een middel voor onweerlegbaarheid. Tot slot kan ook authenticatie en autorisatie worden ondersteund.

Het betekent wel dat er een extra inspanning gele-

verd moet worden om deze manier van beveiliging voor elkaar te boksen. Denk hierbij alleen al aan de extra ontwikkelinspanning. Tevens kan deze methode performancetechnisch een extra belasting geven.

In de inleiding hebben we gezien dat om een aantal redenen de beveiliging van een service-architectuur extra aandacht vraagt: de service-consumers en benodigde providers zijn vooraf onbekend, er wordt gewerkt in een heterogene omgeving, bestaande authenticatie en autorisatie kan niet zonder meer worden hergebruikt en het externe karakter met onbekende partijen brengen ook zo hun uitdagingen met zich mee.

We hebben gezien dat de methode van beveiliging op berichtniveau (uiteraard naast allerlei andere, gangbare methoden om een netwerk of systeem te beveiligen!) een goede oplossing biedt om diverse bedreigingen te lijf te gaan. Voor deze manier van beveiliging zijn specificaties en standaarden beschikbaar of daar wordt hard aan gewerkt. De extra inspanning van beveiliging zal altijd moeten worden afgewogen tegen het risico dat je ermee wegneemt.

Daarnaast blijft de constatering, dat er een spanningsveld bestaat tussen enerzijds de flexibiliteit die een servicesarchitectuur beoogt via bijvoorbeeld de losse koppeling, en anderzijds de veiligheid dat gevoelige gegevens niet zomaar op straat komen te liggen. Dit spanningsveld maakt dat het beveiligen van een servicegeoriënteerde omgeving een uitdaging is, zelfs voor ervaren security-architecten.

Hein Vader is managing consultant bij Capgemini, en is senior software engineer en software architect. Loek Bakker is als senior consultant werkzaam bij Capgemini, en is gespecialiseerd in architectuur en integratievraagstukken.