Beveiliging was ooit voornamelijk een infrastructuur-issue: een systeem afschermen, een firewall dichtzetten en een draadloos netwerk controleren. Met de komst van gedistribueerde applicaties die overal communiceren moeten de applicatie-architect en -ontwikkelaar zich meer dan ooit bewust worden van de gaten en hoe ze te dichten.

Beveiligingniveaus in Windows Communication Foundation

Configureerbaar loodgieterswerk

.NET Framework 3.0, voorheen WinFX, bevat Windows Communication Foundation: de nieuwe manier om geavanceerde gedistribueerde systemen te maken. Wanneer je deze nieuwe technologieën gebruikt, moeten ook bestaande beveiligingsuitdagingen worden overwonnen. Gelukkig is dat gemakkelijker geworden dan voorheen. Maar voordat je specifieke beveiligingsuitdagingen kunt aanpakken, moet je je bewust zijn van de mogelijkheden en opties die er zijn. Laten we daarom eens beter kijken naar WCF en beveiliging.

Windows Communication Foundation

WCF heeft alle goede eigenschappen van webservices, System.EnterpriseServices, remoting, System.Messaging en WSE. Deze technologieën hebben ieder hun sterke punten, zoals ondersteuning van transacties, betrouwbaarheid en beveiliging. Ze hebben echter weinig gedeelde code of configuratie-syntax. Om een techniek, bijvoorbeeld EnterpriseServices, te vervangen door een andere, bijvoorbeeld remoting, is werkelijk een kwestie van het herschrijven van interfaces, communicatiepatronen en uitgewisselde data. Ook is het aanpassen van protocollen, gedragingen en beveiliging vaak lastig of niet eens mogelijk.

Zou het niet aardig zijn als dit kon worden gedaan door configuratie, zonder ontwikkeling en hercompileren? Dat is nu precies wat WCF doet. Het biedt een eenduidig programmeermodel, vermindert 'loodgieten' en biedt meer tijd voor het ontwikkelen van bedrijfsfunctionaliteit. Microsoft heeft bestaande technieken gecombineerd en een nieuwe manier gevonden voor het definiëren en configureren van services. Ik zeg configureren, omdat dat is wat je zou moeten doen met loodgieten. Als je een beveiligd protocol wil, heb je het; als je encryptie wilt, kan dat met één klik gerealiseerd worden.

Om met een WCF-service te kunnen praten of er een te maken, heb je een endpoint nodig, die vaak het ABC van WCF genoemd worden. Een endpoint bestaat uit een Adres, een Binding en een Contract. Daarnaast heb je bestanden, SDK's en een framework nodig om het te laten werken. Begin met het downloaden van .NET Framework 3.0.[1] Er zijn meer plaatsen waar je tutorials kunt vinden voor het maken van WCF-services. MSDN en andere specifieke sites zouden je eerste halte moeten zijn als je op zoek bent naar naslag informatie over Windows Communication Foundation[2]

Applicatiebeveiliging Beveiliging is een gevoelig onderwerp: de belangen kunnen hoog zijn en het speelveld is enorm. Als je aan beveiliging denkt, komen er veel zaken naar boven: Gegevens, applicaties, netwerken, fysieke beveiliging en procedures.

Zelfs applicatiebeveiliging is een breed gebied, dat reikt van het voorkomen van cross-site scripting en sql-injectie tot 'man-in-the-middle' en denial-of-service aanvallen. Ik zal niet in dit gehele gebied duiken. Als je meer wilt weten over webservice-beveiliging, kijk dan op de MSDN-website[3]. In dit artikel wordt ervan uitgegaan dat de informatie die wordt overgebracht geldig en

authentiek is. Er blijven echter nog genoeg vragen onbeantwoord: hoe beveilig je je bericht en het vervoer?

Beveiliging gaat in het algemeen over autorisatie en authenticatie: Is de gebruiker wie hij zegt te zijn en mag hij de gevraagde operatie uitvoeren?

```
[PrincipalPermission(SecurityAction.Demand,
Role="Manager")]
public bool Vote()
{
    if (Thread.CurrentPrincipal.IsInRole
    ("SeniorManager"))
    {
        // Do something only senior managers
           are allowed to do
    }
}
```

Het bovenstaande Code Access Security voorbeeld toont twee manieren om beveiligingsmogelijkheden van het .NET Framework toe te passen: rechten uitvragen door het gebruik van attributen en de IsInRole methode van het principal object. Ook heb je informatie nodig om het systeem te bewaken en problemen aan te pakken. Voortdurend slaat het systeem de identiteit van de gebruiker op en beveiligingsgebeurtenissen zodat later het toelaten en tegenhouden van bewerkingen door gebruikers bewezen kan worden.

Als je gedistribueerde applicaties gaat maken, is er nog een andere beveiligingsdimensie: integriteit en vertrouwelijkheid. Omdat je systeem gegevens aan het versturen is over een netwerk, een bedrijfsnetwerk of het wereldwijde web, wil je dat de gegevens aankomen zoals je ze verstuurd hebt en wellicht wil je ook dat anderen het bericht niet hebben kunnen lezen. Al deze zaken lijken vrij eenvoudig als je ze leest, maar hun beschrijving en betekenis kunnen veel gevolgen hebben voor de technische oplossing die je gaat leveren.

Beveiliging in WCF De basisfunctionaliteit die je verwacht van WCF is het versturen en ontvangen van berichten, SOAP-berichten om exact te zijn. WCF beveiligt deze berichten. WCF beveiligt ook toegang naar andere bronnen. Het doel van WCF is het leveren van beveiliging die eenvoudig en interoperabel is. Bovendien kun je, indien gewenst, je eigen WCF-beveiliging bouwen en uitbreiden, zodat het aan jouw eisen voldoet. Het WCF-beveiligingsmodel is gebaseerd op inloggegevens die toegang geven tot een bepaalde zaak. Dit is standaardgedrag: Van alle beschikbare bindings controleert alleen basicHttpBinding geen inloggegevens. Alle beveiligingscontroles zijn gebaseerd op open standaarden die toegepast worden in de IT-industrie. WsHttpBinding bijvoorbeeld gebruikt protocollen als WS-Addressing, WS-SecureConversation en WS-Trust. Zie WS-specificaties voor beveiliging als de interop laag, zodat je interne beveiliging kunt uitbreiden naar ieder ander platform.

Inloggegevens en verklaringen Tot nu toe heb je gelezen over verklaringen en inloggegevens, maar wat zijn dat precies? Een verklaring is een uitspraak over een instantie gedaan door een andere instantie. Dit klinkt wat vaag, dus laten we een praktisch voorbeeld bekijken: Neem een vorm van identificatie zoals een paspoort of een rijbewijs. Hierop staat je naam, je geboortedatum, en een nummer. Deze vorm van identificatie is uitgegeven door de overheid of een andere instantie die algemeen wordt vertrouwd. De foto bewijst dat jij degene bent die beschreven is. Je was al bekend met de term 'Code Access Security', nu is het tijd om 'Claims Based Access Security' (CBAS) te introduceren. De inloggegevens bevatten een aantal verklaringen die de instantie beschrijven in de digitale wereld. Dit is je beveiligingsbewijs dat over processen en machines reist.

Inloggegevens worden gebruikt om gebruikers te autoriseren. WCF ondersteunt de volgende authenticatiewaarden: None, Username (en wachtwoord), Windows en X.509. Deze waarden kunnen gebruikt worden voor het credentialType attribuut in WCF-configuratie. None is een expliciete keuze om anonieme toegang toe te staan en Windows en X.509 zijn de veilige waarden. De username en wachtwoord-instelling lijken niet erg veilig, maar dit kan gebruikt worden door een ASP.NET Membership Provider om andere gegevensbronnen te gebruiken. Als een credentialtype gespecificeerd is, dan moeten zowel de service als de client hun respectievelijke inloggegevens gebruiken.

Beveiliging op transportniveau In een webomgeving is HTTPS het eerste wat in gedachten komt bij beveiligde verbinding. Internet Information Services maakt HTTPS erg gemakkelijk en gebruikers verwachten dat HTTPS gebruikt wordt als er gevoelige informatie wordt uitgewisseld. Webformulieren die HTTPS gebruiken komen veel voor en een van de voordelen is de relatieve goede performance. Een eis voor zo'n scenario is de beschikbaarheid van een webserver die HTTPS ondersteund. Een mogelijk nadeel is dat, zoals je je kunt voorstellen, de S van HTTPS verdwijnt zodra HTTP niet meer beschikbaar is. Het bericht wordt veilig verstuurd, maar zodra het bij de ontvangende partij aankomt, verdwijnt de beveiliging en kunnen de gegevens binnen de service onderschept worden of verder verstuurd worden. Toch kan deze beveiligingsvorm voldoende zijn voor veel scenario's. Andere transport client credential types kunnen none, basic, digest, ntlm

en windows zijn. Je kunt beveiliging op transportniveau activeren door de binding aan te geven in het configuratiebestand:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.serviceModel>
        <services>
            <service type="Server.
            WcfService">
                <endpoint
                    address="your service
                    address"
                    binding="basicHttp
                    Binding"
                    bindingConfiguration=
                   "SecureBasicHttpBinding"
                    contract="Server.
                    IWcfService"
                />
            </service>
        </services>
        <bindings>
            <basicHttpBinding>
                <binding name="SecureBasic
                HttpBinding">
                 <securitymode="Transport">
                        <transport client
CredentialType="None"/>
                    </security>
                </binding>
            </basicHttpBinding>
        </bindings>
    </system.serviceModel>
</configuration>
```

Voordat je de volgende configuratieinstellingen gaat kopiëren, is het ook belangrijk de Service Configuration Editor Tool te bekijken, die in de SDK zit. Deze tool helpt bij het maken van configuratiebestanden door bepaalde items te selecteren in een grafische omgeving. De standaard installatielocatie is C:\Program Files\Microsoft SDKs\Windows\v1.0\bin\SvcConfigEditor.exe.

Beveiliging op berichtniveau Het overduidelijke alternatief voor beveiliging op transportniveau is berichtbeveiliging. WCF ondersteunt transportbeveiliging en biedt ook beveiliging voor SOAP-berichten, waar het belangrijkste is dat er van het begin tot einde kan worden beveiligd. Met beveiliging op berichtniveau kan je een bericht versleutelen zodra het een vertrouwde omgeving verlaat en het ontsleutelen zodra het op het eindpunt aankomt. Dat kan meerdere stations verder zijn. Message client credential types kunnen none, windows, username, certificate of CardSpace (InfoCard) zijn [4]. Berichtbeveiliging is flexibeler en stelt je in staat op bepaalde gevoelige onderdelen van het bericht te versleutelen. Het nadeel van berichtbeveiliging op dit moment is het gebruik, wat gestandaardiseerd wordt, en de performance hit die je krijgt door het verpakken en uitpakken van het bericht.

```
<wsHttpBinding>
    <binding>
        <security mode="Message">
            <message clientCredentialType=
            "Windows">
        </security>
    </binding>
</wsHttpBinding>
```

Gemengde beveiliging Vroeger moest je kiezen tussen transportbeveiliging of berichtbeveiliging. Nu is er een derde optie: TransportWithMessageCredential of gemengde beveiliging. Deze optie gebruikt de transportlaag om de berichtoverdracht te beveiligen, terwijl ieder bericht de inloggegevens bevat die gebruikt kunnen worden door de services. Dit combineert het performancevoordeel van transportbeveiliging met het voordeel van rijke inloggegevens van berichtbeveiliging. Dit is beschikbaar in de volgende bindings: BasicHttpBinding, WSFederationHttpBinding, NetTcpBinding, en WSHttpBinding.

```
<wsHttpBinding>
    <binding>
        <security mode="TransportWithMessage
        Credential">
            <message clientCredentialType=
            "Windows"/>
        </security>
    </binding>
</wsHttpBinding>
```

Let erop dat er ook een Both mode bestaat die eist dat authenticatie zowel bij de service als bij de client gebeurt, maar dat kan alleen worden gebruikt in combinatie met netMsmqBinding.

Bindings Bindings zijn belangrijke zaken in de WCF-infrastructuur: Met bindings kun je een of meerdere WS-* protocollen, codering en transport aangeven die je wilt ondersteunen. De codering beschrijft hoe berichten worden geserialiseerd. Transport bepaalt het protocol dat gebruikt wordt voor communicatie. Dit wordt duidelijker als je naar een paar kijkt.

We hebben BasicHttpBinding al gezien. Deze binding gebruikt WS-I basic profile protocol, XML-codering en HTTP voor het transport. Deze binding is geschikt voor

interop scenario's, maar zoals eerder genoemd, implementeert deze geen standaardbeveiliging. Als je berichtenbeveiliging wilt, gebruik je WSHttpBinding of WSDualHttpBinding. Er is een verschil in de patronen van berichtuitwisseling. In dit artikel wordt dat niet verder uitgewerkt. Beide gebruiken ook XML voor codering en HTTP voor transport.

Dan zijn er ook nog bindings die binaire codering en transportbeveiliging gebruiken. Om dit te laten werken moet Windows Communication Foundation zowel de zender als de ontvanger zijn. Deze eis beperkt het gebruik van Net*Bindings maar het biedt waardevolle optimalisaties. Vervang de ster in Net*Bindings door respectievelijk Tcp, NamedPipe of Msmq en je weet het transportprotocol dat ze gebruiken. Logischerwijs moet NetMsmqBinding gebruikt worden in asynchrone scenario's omdat het leunt op message queuing, waar NetTcpBinding en NetNamedPipeBinding betrouwbare sessies en transacties ondersteunen.

De bindingsstructuur is open. Gebruik een standaardbinding als het kan, maar je kunt eenvoudig een binding uitbreiden of zelf een binding maken. Gebruik hiervoor de CustomBinding class uit de System.ServiceModel. Channels namespace, bepaal de beveiliging, codering en transport en je hebt je eigen binding.

Controleren Windows Communication Foundation ondersteunt auditing. Dit betekent dat interessante gebeurtenissen kunnen worden opgeslagen en later worden nagekeken. Beveiligingsgebeurtenissen worden in het eventlog opgeslagen door het gebruik van ServiceSecurityAuditBehavior uit de System. ServiceModel.Description namespace.

```
ServiceSecurityAuditBehavior audit = new
      ServiceSecurityAuditBehavior();
audit.AuditLogLocation = AuditLogLocation.
Application;
```

Deze behavior staat je ook toe om berichtauthenticatie en auditniveaus te configureren. Ook kun je dit gedrag instellen in het configuratiebestand van de service.

```
<behavior name="auditBehavior">
    <serviceSecurityAudit
        auditLogLocation="Application"
        suppressAuditFailure="true"
        serviceAuthorizationAuditLevel=
        "Failure"
        messageAuthenticationAuditLevel=
        "Success, Failure" />
</behavior>
```

Een meer technische aanpak van loggen is het activeren van tracing. Misschien ken je tracing van ASP.NET waar je details kunt zien van het runtime creëren van een pagina. In WCF kan je dit met een paar configuratieregels aanzetten en je laten informeren door een bestand met waardevolle informatie om een EHBO bij een service te kunnen toepassen.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.diagnostics>
        <sources>
            <source
                name="System.ServiceModel.
                MessageLogging"
                switchValue="Verbose">
                <listeners>
                    <add name="xml"
                    type="System.
Diagnostics.XmlWriterTraceListener"
                    initializeData=
                    "c:\message.log" />
                </listeners>
            </source>
        </sources>
        <trace autoflush="true" />
    </system.diagnostics>
    <system.serviceModel>
        <diagnostics>
            <messageLogging
            logEntireMessage="true"
                maxMessagesToLog="300"
                logMessagesAtServiceLevel=
                "false"
                logMalformedMessages="true"
                logMessagesAtTransportLevel=
                "true" />
        </diagnostics>
        <services>
            <service type="Server.
            WcfService">
                <endpoint address=
                "your service address"
                    binding="basicHttp
                    Binding"
                    contract="Server.
                    IWcfService"/>
            </service>
        </services>
    </system.serviceModel>
</configuration>
```

Dit is een goed moment om te kijken naar de trace viewer tool die in de SDK zit. Deze tool kan een grafische weergave tonen van de gegevens die gelogd zijn door tracing. De standaard-installatielocatie is C:\Program Files\Microsoft SDKs\Windows\v1.0\bin\SvcTraceViewer.exe. Als je specifieke eisen hebt voor

logging dan is dat geen probleem. Met behulp van een betrouwbaar logging-mechanisme, zoals Enterprise Library van Microsoft Patterns & Practices, kan je je eigen logging-strategie implementeren. In webomgevingen heb je een HttpContext die informatie bevat over de ingelogde identiteit. In WCF servicescenario's kan je OperationContext en ServiceSecurityContext gebruiken voor deze informatie.

Conclusie Windows Communication Foundation is een grote stap vooruit op het gebied van ontwerpen, ontwikkelen en onderhouden van services. Het is gebouwd op het .NET Framework. Daarnaast kun je de bekende Visual Studio 2005 omgeving gebruiken om services te maken die de authenticatie, autorisatie, integriteit, vertrouwelijkheid en auditing bevatten. WCF geeft je configureerbaar loodgieterswerk en richt je op het creëren van bedrijfswaarde op een betrouwbare, transactionele en vooral veilige manier.

Referenties

[1] http://msdn.microsoft.com/windowsvista/downloads/getthebeta/default.aspx
[2] http://www.netfx3.com
[3] http://msdn.microsoft.com/practices/default.aspx?pull=/library/en-us/dnpag2/html/wssp.asp
[4] http://msdn.microsoft.com/winfx/reference/InfoCard/default.aspx

Voorbeelden zijn gebaseerd op de WinFX February CTP versie.

Pieter de Bruin is consultant bij Avanade, een joint-venture van Accenture en Microsoft. (pieterd@avanade.com).

Patches PATCHES Patches PATCHES Patches PATCHES

Artikelen over onderwerpen als software-ontwikkeling, Java, UML, eXteme Programming en nog veel meer vindt u in het Online Archief van Array Publications. Vaktijdschriften als Software Release, Java Magazine, Database Magazine en ons Oracle vakblad Optimize hebben hun artikelenarchief online gezet. Dankzij de heldere zoekstructuur vindt u snel wat u zoekt op www.release.nl.

Microsoft maakt winnaars Imagine Cup 2006 bekend

Wat begon met 65.000 studenten uit meer dan 100 landen, bereikte deze week zijn climax: Microsoft heeft de winnaars van de Imagine Cup 2006 bekendgemaakt. Het team van de Italiaanse Politecnico di Torino - Giorgio Sardo, Massimo Paternoster, Silvia Perrone en Andrea Sossich - zegevierde in de categorie softwareontwerp en won daarmee 25.000 dollar. De Imagine Cup, Microsofts belangrijkste competitie voor technologiestudenten, stimuleert studenten over de hele wereld tot het bedenken van creatieve en technologische innovaties. De winnaar van de Nederlandse versie van de Imagine Cup, een studententeam uit Twente, heeft de finale helaas niet gehaald.

"Het Nederlandse team heeft het helaas niet gehaald, maar twee van de teamleden willen graag volgend jaar weer meedoen", zegt Maarten Jan Vermeulen, Academic Relations Manager bij de Developer & Platform Group van Microsoft Nederland. "Dit jaar deden er drie Nederlandse teams mee aan de Imagine Cup waarvan er één doorging naar India. Maar voor volgend jaar willen we graag veel meer deelnemers aantrekken en meer focus leggen op andere categorieën dan softwareontwerp, 'korte film' bijvoorbeeld. Daarom is de werving van teams al begonnen en hebben we al reacties gekregen van geïnteresseerden."

De 42 teams van softwarefinalisten gebruikten Microsoft-technologie en Microsoft.NET-webdiensten om een applicatie te ontwikkelen die aansluit op het thema van de Imagine Cup: stel je een wereld voor waarin technologie een gezonder leven mogelijk maakt. Uiteindelijk werden er 181 studenten uit 72 teams en 42 landen uitverkoren om in de finales van de Imagine Cup ten strijde te trekken, in 6 categorieën: software, algoritme, IT, korte film, interface en de programmeerwedstrijd Hoshimi. De teams moesten, afhankelijk van de categorie waarin ze uitkwamen, een reeks uitdagingen op het gebied van multimedia of technologie aangaan.

De Imagine Cup, die dit jaar voor de vierde keer werd gehouden, is een wedstrijd die dient als uitlaatklep voor studenten om hun technologische en artistieke gaven buiten de collegezaal te tonen. Het daagt studenten uit om hun intelligentie, creativiteit en energie in dienst te stellen van een betere wereld. Het geeft deelnemers de kans een rol te spelen in de toekomst van technologie, software en computers. De teams ontwikkelen innovatieve, praktisch uitvoerbare projecten en bedenken échte oplossingen voor échte problemen.

De Imagine Cup 2007 wordt gehouden in Seoul, Korea. Meer informatie is te vinden op www.imaginecup.com.