Kapitel 11 – .NET Remoting-Sicherheit

Erstellen sicherer ASP.NET-Anwendungen

Authentifizierung, Autorisierung und sichere Kommunikation

J.D. Meier, Alex Mackman, Michael Dunner und Srinath Vasireddy

Microsoft Corporation

Oktober 2002

Auf der Orientierungsseite finden Sie einen Ausgangspunkt und eine vollständige Übersicht zum *Erstellen sicherer ASP.NET-Anwendungen*.

Zusammenfassung

In diesem Kapitel wird gezeigt, wie sichere .NET Remoting-Lösungen implementiert werden.

Inhalt

Das .NET Framework stellt eine Remoteinfrastruktur bereit, die Clients die Kommunikation mit Objekten ermöglicht, die sich in Remoteanwendungsdomänen und -prozessen oder auf Remotecomputern befinden. In diesem Kapitel wird gezeigt, wie sichere .NET Remoting-Lösungen implementiert werden.

.NET Remoting-Architektur

In Abbildung 11.1 wird die .NET Remoting-Standardarchitektur veranschaulicht, bei der ein Remoteobjekt in ASP.NET verwaltet wird. Ein ASP.NET-Host stellt zusammen mit dem HTTP-Kanal für die Kommunikation den empfohlenen Ansatz dar, wenn die Sicherheit das Hauptanliegen darstellt, da es dadurch dem Remoteobjekt ermöglicht wird, die zugrunde liegenden Sicherheitsdienste zu nutzen, die von ASP.NET und IIS bereitgestellt werden.

Weitere Informationen zum Bereich möglicher Host- und Kanaltypen finden Sie zusammen mit Vergleichsinformationen weiter unten in diesem Kapitel unter "Auswählen eines Hostprozesses".

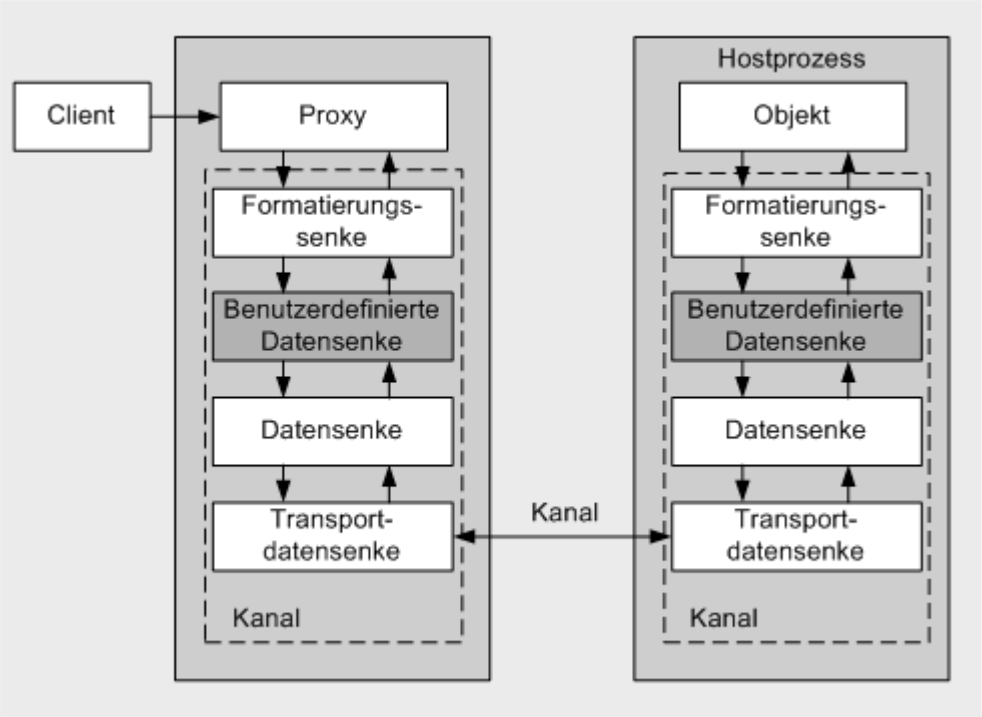

Abbildung 11.1
.NET Remoting-Architektur

Der Client kommuniziert mit einem prozessinternen Proxyobjekt. Die Anmeldeinformationen (z. B. Benutzernamen, Kennwörter, Zertifikate usw.) können für die Authentifizierung über den Remoteobjektproxy festgelegt werden. Der Methodenaufruf durchläuft eine Reihe von Senken (Sie können eigene benutzerdefinierte Senken implementieren, z. B. zum Durchführen der Datenverschlüsselung) zu einer Transportsenke, die für das Senden der Daten über das Netzwerk verantwortlich ist. Auf der Serverseite wird der Aufruf durch dieselbe Pipeline geleitet, nach der der Aufruf zum Objekt gesendet wird.

Hinweis: Der in diesem Kapitel verwendete Begriff *Proxy* bezieht sich auf das clientseitige, prozessinterne Proxyobjekt, über das Clients mit dem Remoteobjekt kommunizieren. Verwechseln Sie dies nicht mit dem Begriff *Proxyserver*.

Remoting-Senken

Das .NET Remoting verwendet Transportkanalsenken, benutzerdefinierte Kanalsenken und Formatierungskanalsenken, wenn ein Client einen Methodenaufruf für ein Remoteobjekt startet.

Transportkanalsenken

Transportkanalsenken übergeben Methodenaufrufe zwischen dem Client und dem Server über das Netzwerk. .NET stellt die Klassen **HttpChannel** und **TcpChannel** bereit, obwohl die Architektur umfassend erweitert werden kann und Sie eigene benutzerdefinierte Implementierungen einsetzen können.

- **HttpChannel** – Dieser Kanal ist für den Einsatz beim Verwalten eines Remoteobjekts in ASP.NET gedacht. Der Kanal verwendet das HTTP-Protokoll und sendet Nachrichten zwischen Client und Server.
- **TcpChannel** – Dieser Kanal ist für den Einsatz beim Verwalten eines Remoteobjekts in einem Dienst oder in einer anderen ausführbaren Datei des Betriebssystems Microsoft® Windows® gedacht. Der Kanal verwendet TCP-Sockets, um Nachrichten zwischen dem Client und dem Server zu senden.
- **Benutzerdefinierte Kanäle** – Ein benutzerdefinierter Transportkanal kann ein beliebiges zugrunde liegendes Transportprotokoll zum Senden von Nachrichten zwischen dem Client und dem Server verwenden. Ein benutzerdefinierter Kanal kann z. B. Named Pipes oder Mailslots verwenden.

Vergleichen von Transportkanalsenken

In der folgenden Tabelle werden die zwei hauptsächlichen Transportkanalsenken verglichen.

Tabelle 11.1: *Vergleich von TcpChannel und HttpChannel*

Feature	TCP-Kanal	HTTP-Kanal	Kommentare
Authentifizierung	Nein	Ja	Der HTTP-Kanal verwendet die von IIS und ASP.NET bereitgestellten Authentifizierungsfunktionen, obwohl die Passport- und Formularauthentifizierung nicht unterstützt werden.
Autorisierung	Nein	Ja	Der HTTP-Kanal unterstützt die von IIS und ASP.NET bereitgestellten Autorisierungsfunktionen. Diese umfassen die NTFS-Berechtigungen, die URL-Autorisierung sowie die Dateiautorisierung.
Sichere Kommunikation	Ja	Ja	Verwenden Sie IPSec mit dem TCP-Kanal. Verwenden Sie SSL und/oder IPSec mit dem HTTP-Kanal.

Benutzerdefinierte Senken

Benutzerdefinierte Kanalsenken können innerhalb der Kanalsenkenpipeline an verschiedenen Stellen verwendet werden, um die zwischen dem Client und dem Server gesendeten Nachrichten zu ändern. Ein Beispiel für eine benutzerdefinierte Kanalsenke stellt eine Kanalsenke dar, die die Verschlüsselung und Entschlüsselung bereitstellt.

Formatierungssenken

Formatierungssenken nehmen Methodenaufrufe an und serialisieren diese in einen Datenstrom, der über das Netzwerk gesendet werden kann. .NET stellt zwei Formatierungssenken bereit:

- **Binäre Formatierung** – Diese verwendet die Klasse **BinaryFormatter**, um Methodenaufrufe in einen serialisierten Binärdatenstrom zu packen, der anschließend (mithilfe von HTTP POST) bereitgestellt wird, um die Daten an den Server zu senden. Die binäre Formatierung legt den Inhaltstyp in der HTTP-Anforderung auf **application/octet-stream** fest.

  Die binäre Formatierung bietet im Vergleich zur SOAP-Formatierung eine überragende Leistung.
- **SOAP-Formatierung** – Diese verwendet die Klasse **SoapFormatter**, um Methodenaufrufe in einer SOAP-Nachricht zu verpacken. Der Inhaltstyp wird in der HTTP-Anforderung auf **text/xml** festgelegt und unter Verwendung von HTTP POST an den Server gesendet.

Aufbau einer Anforderung beim Hosting in ASP.NET

Die Endpunkte von Remoteobjekten werden durch URLs angegeben, die mit der Dateierweiterung REM oder SOAP enden, z. B. **http://BeliebigerServer/vDir/Remoteobjekt.soap**. Wenn eine Anforderung für ein Remoteobjekt (mit der Erweiterung REM oder SOAP) von IIS empfangen wird, erfolgt deren Zuordnung (innerhalb von IIS) zur ASP.NET ISAPI-Erweiterung (**Aspnet_isapi.dll**). Die ISAPI-Erweiterung leitet die Anforderung an eine Anwendungsdomäne im ASP.NET-Workerprozess (**Aspnet_wp.exe**) weiter. Die Abfolge der Ereignisse ist in Abbildung 11.2 dargestellt.

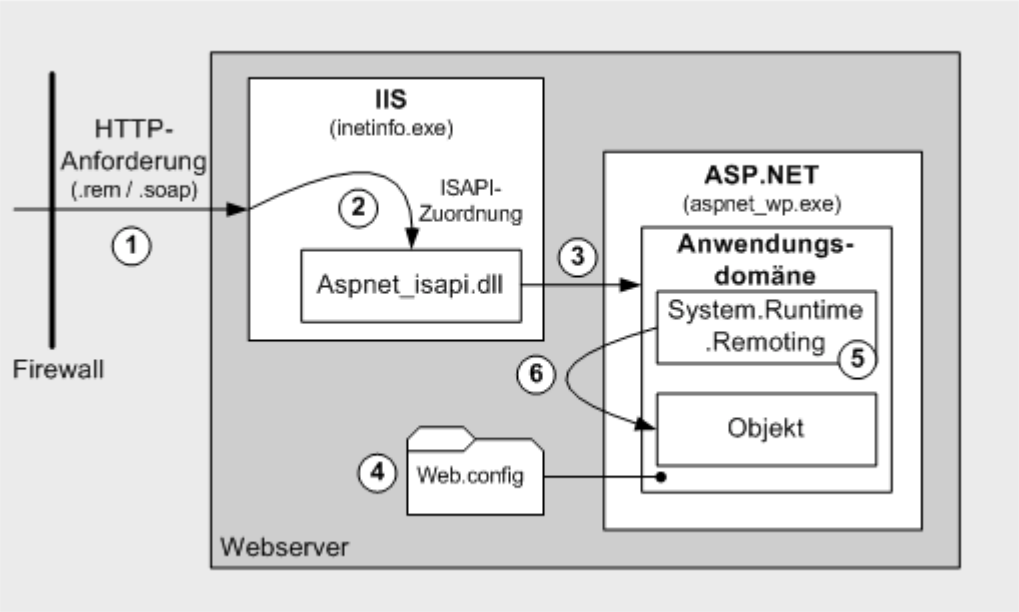

Abbildung 11.2
Serverseitige Verarbeitung

Abbildung 11.2 zeigt die folgende Ereignisabfolge:

1. Über HTTP wird eine SOAP- oder REM-Anforderung empfangen und einem bestimmten virtuellen Verzeichnis auf dem Webserver zugeordnet.
2. IIS überprüft die SOAP-/REM-Zuordnung und ordnet die Dateierweiterung der ASP.NET ISAPI-Erweiterung, **Aspnet_isapi.dll**, zu.

3. Die ISAPI-Erweiterung überträgt die Anforderung an eine Anwendungsdomäne im ASP.NET-Workerprozess (**Aspnet_wp.exe**). Wenn es sich hierbei um die erste Anforderung handelt, die an diese Anwendung gesendet wurde, wird eine neue Anwendungsdomäne erstellt.
4. Der Handler **HttpRemotingHandlerFactory** wird aufgerufen, und die Remoteinfrastruktur liest den Abschnitt <**system.runtime.remoting**> der Datei **Web.config**, der die serverseitige Objektkonfiguration (z. B. Einzelaufruf- oder Singleton-Parameter) und die Autorisierungsparameter (des Elements <**authorization**>) steuert.
5. Die Remoteinfrastruktur stellt die Position der Assembly fest, die das Remoteobjekt enthält, und erstellt eine Instanz von dieser Assembly.
6. Die Remoteinfrastruktur liest die HTTP-Header und den Datenstrom und ruft dann die Methode für das Remoteobjekt auf.

Hinweis: Während dieses Prozesses ruft ASP.NET die übliche Folge von Ereignishandlern auf. Sie können optional eine oder mehrere dieser Ereignishandler in der Datei **Global.asax** implementieren, z. B. **BeginRequest**, **AuthenticationRequest**, **AuthorizeRequest** usw. Wenn die Anforderung die Remoteobjektmethode erreicht, wird das Objekt **IPrincipal**, das den authentifizierten Benutzer darstellt, in **HttpContext.User** (und **Thread.CurrentPrincipal**) gespeichert und steht dort für die Autorisierung zur Verfügung, beispielsweise durch Verwenden von PrincipalPermission-Forderungen und programmatischen Rollenüberprüfungen.

ASP.NET und der HTTP-Kanal

Das Remoting besitzt kein eigenes Sicherheitsmodell. Die Authentifizierung und Autorisierung zwischen dem Client (Proxy) und dem Server (Remoteobjekt) werden vom Kanal- und Hostprozess durchgeführt. Sie können die folgende Kombination aus Hosts und Kanälen verwenden:

- **Eine benutzerdefinierte ausführbare Datei und der TCP-Kanal -** Diese Kombination stellt keine integrierten Sicherheitsfunktionen bereit.
- **ASP.NET und der HTTP-Kanal -** Diese Kombination stellt die Authentifizierung und Autorisierung über die zugrunde liegenden Sicherheitsfunktionen von ASP.NET und IIS bereit.

Objekte, die in ASP.NET verwaltet werden, profitieren von den zugrunde liegenden Sicherheitsfunktionen von ASP.NET und IIS. Dabei handelt es sich um die Folgenden:

- **Authentifizierungsfunktionen** – Die Windows-Authentifizierung ist in **Web.config** konfiguriert.

  ```
  <authentication mode="Windows"/>
  ```

  Die Einstellungen in IIS steuern, welcher HTTP-Authentifizierungstyp verwendet wird.

  Zum Authentifizieren von Anforderungen werden allgemeine HTTP-Header verwendet. Sie können Anmeldeinformationen für den Client bereitstellen, indem Sie den Remoteobjektproxy konfigurieren oder Standardanmeldeinformationen verwenden.

Sie können nicht die Formular- oder Passport-Authentifizierung verwenden, da der Kanal keine Möglichkeit bietet, dem Client den Zugriff auf Cookies zu gewähren. Dies stellt jedoch eine Anforderung für diese beiden Authentifizierungsmechanismen dar. Für die Formular- und Passport-Authentifizierung ist es außerdem notwendig, dass eine Umleitung auf eine Anmeldeseite erfolgt, die eine Interaktion vom Client erfordert. Serverseitige Remoteobjekte wurden für den nicht interaktiven Einsatz entwickelt.

- **Autorisierungsfunktionen** – Clients werden unter Verwendung von ASP.NET-Standardautorisierungsverfahren autorisiert.

  Konfigurierbare Autorisierungsoptionen:

  - URL-Autorisierung
  - Dateiautorisierung (erfordert eine bestimmte Konfiguration, wie weiter unten in diesem Kapitel unter "Verwenden der Dateiautorisierung" beschrieben)

  Programmatische Autorisierungsoptionen:

  - PrincipalPermission-Forderungen (deklarativ und imperativ)
  - Explizite Rollenüberprüfungen unter Verwendung von **IPrincipal.IsInRole**

- **Sichere Kommunikationsfunktionen** – Um den Transport von Daten zwischen dem Client und dem Server zu sichern, sollte SSL (und/oder IPSec) verwendet werden.

Weitere Informationen

- Weitere Informationen zu den von ASP.NET und IIS bereitgestellten Authentifizierungs- und Autorisierungsfunktionen finden Sie in Kapitel 8, "ASP.NET-Sicherheit".
- Weitere Informationen zum Verwalten von Objekten in ASP.NET/IIS finden Sie in der Microsoft Knowledge Base im Artikel Q312107, "HOW TO: Host a Remote Object in Microsoft Internet Information Services" (US).

.NET Remoting-Gatekeeper

Folgende Autorisierungspunkte (oder Gatekeeper) stehen für ein Remoteobjekt zur Verfügung, das von ASP.NET verwaltet wird:

- **IIS** – Bei deaktivierter anonymer Authentifizierung gestattet IIS nur Anforderungen von Benutzern, die entweder in der eigenen oder in einer vertrauenswürdigen Domäne authentifiziert werden können. IIS bietet außerdem das Filtern von IP-Adressen und das Filtern für DNS.
- **ASP.NET**
  - **UrlAuthorizationModule** – Sie können <**authorization**>-Elemente in der Datei **Web.config** der Anwendung konfigurieren, um zu steuern, welche Benutzer und Benutzergruppen Zugriff auf die Anwendung erhalten sollen. Die Autorisierung basiert auf dem Objekt **IPrincipal**, das in **HttpContext.User** gespeichert ist.
  - **FileAuthorizationModule** – **FileAuthorizationModule** steht für Remotekomponenten zur Verfügung, obwohl dies eine bestimmte Konfiguration erfordert, wie weiter unten in diesem Kapitel unter "Verwenden der Dateiautorisierung" beschrieben wird.

Hinweis: Der Identitätswechsel ist für eine funktionierende Dateiautorisierung nicht erforderlich.

Die Klasse **FileAuthorizationModule** führt nur für die angeforderte Datei oder URI (z. B. REM und SOAP) Zugriffsüberprüfungen durch und nicht für Dateien, auf die über Code innerhalb des Remoteobjekts zugegriffen wird.

- **PrincipalPermission-Forderungen und explizite Rollenüberprüfungen** – Zusätzlich zu den konfigurierbaren IIS- und ASP.NET-Gatekeepern können Sie auch PrincipalPermission-Forderungen (deklarativ oder imperativ) als zusätzlichen feinstufigen Zugriffssteuerungsmechanismus verwenden. Principalberechtigungsüberprüfungen ermöglichen es, basierend auf der Identität und Gruppenmitgliedschaft einzelner Benutzer, den Zugriff auf Klassen, Methoden oder einzelne Codeblöcke, wie durch das dem aktuellen Thread zugeordnete Objekt **IPrincipal** definiert, zu steuern.

Hinweis: Zum Anfordern der Rollenmitgliedschaft verwendete Principalberechtigungsüberprüfungen unterscheiden sich vom Aufrufen von **IPrincipal.IsInRole** zum Testen der Rollenmitgliedschaft. Ersteres führt zu einer Ausnahme, wenn der Aufrufer kein Mitglied der angegebenen Rolle ist, während Letzteres einfach einen booleschen Wert zurückgibt, um die Rollenmitgliedschaft zu bestätigen.

Bei der Windows-Authentifizierung hängt ASP.NET automatisch das Objekt **WindowsPrincipal** an, das den authentifizierten Benutzer für die aktuelle Webanforderung (unter Verwendung von **HttpContext.User**) darstellt.

Authentifizierung

Wenn Sie das Remoting gemeinsam mit einem ASP.NET-Webanwendungsclient verwenden, erfolgt die Authentifizierung innerhalb der Webanwendung und auf dem Remoteobjekthost. Die verfügbaren Authentifizierungsoptionen für den Remoteobjekthost hängen vom Hosttyp ab.

Hosting in ASP.NET

Wenn das Hosting für Objekte in ASP.NET erfolgt, wird der HTTP-Kanal verwendet, um Methodenaufrufe zwischen dem clientseitigen Proxy und dem Server zu übertragen. Der HTTP-Kanal verwendet das HTTP-Protokoll, um den Remoteobjektproxy für den Server zu authentifizieren.

In der folgenden Liste wird der Bereich der Authentifizierungsoptionen aufgeführt, die beim Hosting in ASP.NET zur Verfügung stehen:

- **IIS-Authentifizierungsoptionen** – Anonyme Authentifizierung, Standardauthentifizierung, Digestauthentifizierung, integrierte Windows-Authentifizierung und Zertifikatsauthentifizierung.
- **ASP.NET-Authentifizierungsoptionen** – Windows-Authentifizierung oder "Keine" (für Implementierungen benutzerdefinierter Authentifizierungen).

Hinweis: Die Formular- und Passport-Authentifizierungen können von .NET Remoting nicht direkt verwendet werden. Aufrufe von Remoteobjekten wurden als "nicht interaktiv" entworfen. Wenn es sich beim Client des Remoteobjekts um eine .NET-Webanwendung handelt, kann die Webanwendung die Formular- und Passport-Authentifizierung verwenden und Anmeldeinformationen explizit an das Remoteobjekt übergeben. Diese Art von Szenario wird weiter unten in diesem Kapitel unter "Übermitteln des ursprünglichen Aufrufers" ausführlicher behandelt.

Hosting in einem Windows-Dienst

Wenn das Hosting für Objekte in einem Windows-Dienst erfolgt, wird der TCP-Kanal verwendet, um Methodenaufrufe zwischen dem Client und dem Server zu übertragen. Dieser verwendet Übertragungen, die auf Ursprungssockets (Raw Sockets) basieren. Da mit Sockets keine Authentifizierung bereitgestellt wird, gibt es für den Server keine Möglichkeit, den Client zu authentifizieren.

In diesem Szenario muss das Remoteobjekt die benutzerdefinierte Authentifizierung verwenden.

Benutzerdefinierte Authentifizierung

Bei der einfachen benutzerdefinierten Authentifizierung kann das Remoteobjekt eine **Login**-Methode offen legen, die einen Benutzernamen und ein Kennwort akzeptiert. Die Anmeldeinformationen können anhand eines Speichers, einer abgerufenen Rollenliste sowie anhand eines an den Client zurückgesendeten Tokens überprüft werden, das für nachfolgende Anforderungen verwendet werden soll. Wenn das Token auf dem Server abgerufen wird, erfolgt mit diesem die Erstellung eines **IPrincipal**-Objekts (mit Rollen), das in **Thread.CurrentPrincipal** gespeichert wird, wo der Einsatz zu Autorisierungszwecken erfolgt.

Andere Beispiele für benutzerdefinierte Authentifizierungen umfassen das Erstellen einer benutzerdefinierten Transportkanalsenke, die einen prozessübergreifenden Kommunikationskanal verwendet, der die Authentifizierung bereitstellt (z. B. Pipes) oder das Erstellen einer Kanalsenke, die die Authentifizierung anhand der Windows-SSPI (Security Service Provider Interface) durchführt.

Weitere Informationen

- Weitere Informationen zum Verwalten eines Objekts in einem Windows-Dienst finden Sie unter "Verwenden eines Windows-Dienstes als Host für ein Remoteobjekt" im Abschnitt "Referenz" dieses Handbuchs.
- Weitere Informationen zu Senken und Senkenketten finden Sie in der MSDN Library im Abschnitt über das .NET Framework unter "Sinks and Sink Chains" (englischsprachig).
- Weitere Informationen zum Erstellen einer benutzerdefinierten Authentifizierungslösung, die SSPI verwendet, finden Sie im MSDN-Artikel ".NET Remoting Security Solution, Part 1: Microsoft.Samples.Security.SSPI Assembly" unter http://msdn.microsoft.com/library/en-us/dndotnet/html/remsspi.asp (englischsprachig).

Hinweis: Die in diesem Artikel beschriebene Implementierung ist ein Beispiel und kein von Microsoft getestetes und unterstütztes Produkt.

Autorisierung

Wenn Objekte von ASP.NET verwaltet und der HTTP-Kanal für die Kommunikation verwendet wird, kann der Client unter Verwendung der folgenden Mechanismen authentifiziert und die Autorisierung durchgeführt werden:

- URL-Autorisierung
- Dateiautorisierung
- PrincipalPermission-Forderungen (deklarativ und imperativ)
- **IPrincipal.IsInRole**-Überprüfungen im Code

Wenn Objekte in einem Windows-Dienst verwaltet werden, wird vom TCP-Kanal keine Authentifizierung bereitgestellt. Deshalb müssen Sie die benutzerdefinierte Authentifizierung und anschließend die Autorisierung durchführen, indem Sie ein **IPrincipal**-Objekt erstellen und es in **Thread.CurrentPrincipal** speichern.

Sie können dann Ihre Methoden des Remoteobjekts über deklarative Überprüfungen von PrincipalPermission-Forderungen mit Anmerkungen versehen, wie nachfolgend gezeigt:

```
[PrincipalPermission(SecurityAction.Demand,
                     Role="Manager")]
void SomeMethod()
{
}
```

Innerhalb des Codes der Methode Ihres Objekts können auch imperative PrincipalPermission-Forderungen und explizite Rollenüberprüfungen mithilfe von **IPrincipal.IsInRole** verwendet werden.

Verwenden der Dateiautorisierung

Sie können die integrierte Windows-Zugriffssteuerung verwenden, um das Remoteobjekt als Windows-Ressource zu sichern. Ohne die Dateiautorisierung (unter Verwendung von Windows-Zugriffssteuerungslisten) steht Ihnen nur die URL-Autorisierung zur Verfügung.

Sie müssen eine physische Datei mit der Dateierweiterung REM oder SOAP im virtuellen Verzeichnis der Anwendung erstellen, um **FileAuthorizationModule** zum Autorisieren des Zugriffs auf Remoteobjektendpunkte (gekennzeichnet durch REM- oder SOAP-URLs) verwenden.

Hinweis: Die Dateierweiterungen REM und SOAP werden von IIS verwendet, um Anforderungen für Objektendpunkte zur ASP.NET ISAPI-Erweiterung (**Aspnet_isapi.dll**) zuzuordnen. Diese existieren normalerweise nicht als physische Dateien.

So konfigurieren Sie die Dateiautorisierung für .NET Remoting:

1. Erstellen Sie eine Datei mit demselben Namen wie **objectUri** (z. B. **RemoteMath.rem**) im virtuellen Stammverzeichnis der Anwendung.
2. Fügen Sie am Anfang der Datei die folgende Zeile hinzu, und speichern Sie dann die Datei:

   ```
   <%@ webservice class="YourNamespace.YourClass" ... %>
   ```

3. Fügen Sie über den Windows-Explorer eine entsprechend konfigurierte Zugriffssteuerungsliste zur Datei hinzu.

   Hinweis: Sie können **objectUri** aus der Datei **Web.config** abrufen, die zum Konfigurieren des Remoteobjekts auf dem Server verwendet wird. Suchen Sie das Element <**wellknown**>, wie im folgenden Beispiel gezeigt wird:

   ```
   <wellknown mode="SingleCall" objectUri="RemoteMath.rem"
   type="RemotingObjects.RemoteMath, RemotingObjects, Version=1.0.000.000
   Culture=neutral, PublicKeyToken=4b5ae668c251b606"/>
   ```

Weitere Informationen

- Weitere Details zu diesen Autorisierungsmechanismen finden Sie in Kapitel 8, "ASP.NET-Sicherheit".
- Weitere Informationen zu PrincipalPermission-Forderungen finden Sie unter "PrincipalPermission-Forderungen" im Abschnitt "ASP.NET-Autorisierungsoptionen" dieses Handbuchs.

Authentifizierungs- und Autorisierungsstrategien

In vielen Anwendungen, die .NET Remoting verwenden, werden die Remoteobjekte verwendet, um auf der mittleren Ebene der Anwendung Geschäftsfunktionen bereitzustellen; diese Funktionen werden dann von den ASP.NET-Webanwendungen aufgerufen. Diese Anordnung wird in Abbildung 11.3 dargestellt.

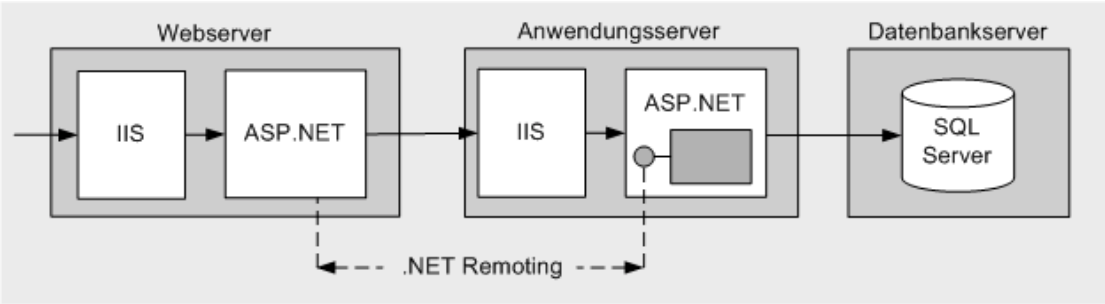

Abbildung 11.3
Von einer ASP.NET-Webanwendung aufgerufene Remoteobjekte

In diesem Szenario können die für die Webanwendung verfügbaren IIS- und ASP.NET-Gatekeeper verwendet werden, um den Zugriff auf den clientseitigen Proxy zu sichern. Die für den ASP.NET-Host auf dem Remoteanwendungsserver zur Verfügung stehenden IIS- und ASP.NET-Gatekeeper stehen zum Sichern des Zugriffs auf das Remoteobjekt bereit.

Es stehen im Wesentlichen zwei Authentifizierungs- und Autorisierungsstrategien für Remoteobjekte zur Verfügung, auf die .NET-Webanwendungen zugreifen.

- Sie können Aufrufer auf dem Webserver authentifizieren sowie autorisieren und dann den Sicherheitskontext des Aufrufers unter Verwendung des Identitätswechsels an das Remoteobjekt übermitteln. Dies ist das Modell mit Identitätswechsel/Delegierung.

  Bei diesem Ansatz verwenden Sie einen IIS-Authentifizierungsmechanismus, der es Ihnen ermöglicht, den Sicherheitskontext des Aufrufers zu delegieren, z. B. die Kerberos-, Standard- oder Formularauthentifizierung (wobei es die beiden letztgenannten Mechanismen ermöglichen, auf die Anmeldeinformationen des Aufrufers zuzugreifen), und die Anmeldeinformationen explizit an das Remoteobjekt zu übermitteln, wobei der Proxy des Remoteobjekts verwendet wird.

  Die von ASP.NET konfigurierbaren und programmatischen Gatekeeper (einschließlich der URL-Autorisierung, der Dateiautorisierung, der PrincipalPermission-Forderungen und .NET-Rollen) stehen zum Autorisieren einzelner Aufrufer innerhalb des Remoteobjekts zur Verfügung.
- Sie können Aufrufer auf dem Webserver authentifizieren sowie autorisieren und dann eine vertrauenswürdige Identität verwenden, um mit dem Remoteobjekt zu kommunizieren. Dies ist das Modell mit vertrauenswürdigem Subsystem.

Dieses Modell beruht darauf, dass die Webanwendung Aufrufer authentifiziert und ordnungsgemäß autorisiert, bevor das Remoteobjekt aufgerufen wird. Sämtliche vom Remoteobjekt erhaltenen Anforderungen von der vertrauenswürdigen Identität, die von der Webanwendung geplant wurden, dürfen fortgesetzt werden.

Weitere Informationen

- Weitere Informationen zu den Modellen mit Identitätswechsel/Delegierung und mit dem vertrauenswürdigen Subsystem finden Sie unter "Ressourcenzugriffsmodelle" in Kapitel 3, "Authentifizierung und Autorisierung".
- Weitere Informationen zum Verwenden des Modells mit dem ursprünglichen Aufrufer mit Remoting finden Sie weiter unten in diesem Kapitel unter "Übermitteln des ursprünglichen Aufrufers".
- Weitere Informationen zum Verwenden des Modells mit vertrauenswürdigem Subsystem mit Remoting finden Sie weiter unten in diesem Kapitel unter "Vertrauenswürdiges Subsystem".

Zugreifen auf Systemressourcen

Weitere Informationen zum Zugriff auf Systemressourcen (z. B. auf das Ereignisprotokoll und die Registrierung) über ein von ASP.NET verwaltetes Remoteobjekt finden Sie unter "Zugreifen auf Systemressourcen" in Kapitel 8, "ASP.NET-Sicherheit". Die in Kapitel 8 beschriebenen Ansätze und Einschränkungen gelten auch für von ASP.NET verwaltete Remoteobjekte.

Zugreifen auf Netzwerkressourcen

Wenn Sie über ein Remoteobjekt auf Netzwerkressourcen zugreifen, müssen Sie die Art der Identität bedenken, die zum Antworten auf Netzwerk-Authentifizierungsherausforderungen vom Remotecomputer verwendet wird. Ihnen bieten sich drei Optionen:

- **Prozessidentität (Standardwert)** – Wenn das Hosting in ASP.NET erfolgt, bestimmt die zum Ausführen des ASP.NET-Workerprozesses verwendete und vom Element <**processModel**> in **Machine.config** definierte Identität den Sicherheitskontext, der für den Ressourcenzugriff verwendet wird.

  Wenn das Hosting in einem Windows-Dienst erfolgt, bestimmt die zum Ausführen des Dienstprozesses (mit dem MMC-Snap-In Dienste konfiguriert) verwendete Identität den Sicherheitskontext, der für den Ressourcenzugriff verwendet wird.
- **Feste Dienstidentität** – Beispielsweise eine Dienstidentität, die durch Aufrufen von **LogonUser** erstellt wurde.

  Hinweis: Verwechseln Sie diese Dienstidentität nicht mit der zum Ausführen des Windows-Dienstes verwendeten Identität. Eine feste Dienstidentität verweist auf ein Windows-Benutzerkonto, das speziell zum Zugreifen auf Ressourcen von einer Anwendung erstellt wurde.

- **Identität des ursprünglichen Aufrufers** – Wenn ASP.NET für den Identitätswechsel konfiguriert ist oder der programmatische Identitätswechsel in einem Windows-Dienst verwendet wird.

Ausführliche Informationen zu den relativen Vorzügen dieser Ansätze finden Sie zusammen mit Konfigurationsdetails unter "Zugreifen auf Netzwerkressourcen" in Kapitel 8, "ASP.NET-Sicherheit".

Übergeben von Anmeldeinformationen an Remoteobjekte zur Authentifizierung

Wenn ein Clientprozess ein Remoteobjekt aufruft, erfolgt dies unter Verwendung eines Proxys. Hierbei handelt es sich um ein lokales Objekt, das dieselbe Reihe von Methoden offen legt wie das Zielobjekt.

Angeben von Clientanmeldeinformationen

Wenn das Remoteobjekt in ASP.NET verwaltet wird und für die Windows-Authentifizierung konfiguriert ist, müssen Sie die zum Authentifizieren verwendeten Anmeldeinformationen mithilfe der Eigenschaft für die Anmeldeinformationen des Kanals angeben. Wenn Sie die Anmeldeinformationen nicht explizit festlegen, wird das Remoteobjekt ohne Anmeldeinformationen aufgerufen. Wenn die Windows-Authentifizierung erforderlich ist, führt dies zu einem HTTP-Status 401, d. h. zu einer Antwort auf den verweigerten Zugriff.

Verwenden von "DefaultCredentials"

Wenn Sie die Anmeldeinformationen des Prozesses verwenden möchten, der den Remoteobjektproxy (oder das aktuelle Threadtoken, wenn der Thread, der den Proxy aufruft, einen Identitätswechsel durchführt) verwaltet, sollten Sie die Eigenschaft für die Anmeldeinformationen des Kanals auf den Wert **DefaultCredentials** festlegen, der vom Anmeldeinformationscache des Prozesses gepflegt wird.

Sie können die Verwendung von **DefaultCredentials** entweder in einer Konfigurationsdatei oder programmgesteuert festlegen.

Explizite Konfiguration

In der Konfigurationsdatei der Clientanwendung (**Web.config**, wenn die Clientanwendung eine ASP.NET-Webanwendung ist) legen Sie für das **useDefaultCredentials**-Attribut des Elements <**channel**> den Wert **True** fest, um anzugeben, dass der Proxy **DefaultCredentials** verwenden soll, wenn er mit dem Remoteobjekt kommuniziert.

```
<channel ref="http" useDefaultCredentials="true" />
```

Programmatische Konfiguration

Zur programmatischen Konfiguration verwenden Sie den folgenden Code, um die Verwendung von **DefaultCredentials** programmatisch einzurichten:

```
IDictionary channelProperties;
channelProperties = ChannelServices.GetChannelSinkProperties(proxy);
channelProperties ["credentials"] = CredentialCache.DefaultCredentials;
```

Verwenden von bestimmten Anmeldeinformationen

Wenn Sie beim Aufrufen eines Remoteobjekts einen bestimmten Satz Anmeldeinformationen für die Authentifizierung verwenden möchten, deaktivieren Sie die Verwendung von Standardanmeldeinformationen in der Konfigurationsdatei, indem Sie die folgende Einstellung verwenden:

```
<channel ref="http" useDefaultCredentials="false" />
```

Hinweis: Die programmatischen Einstellungen setzen immer die Einstellungen in der Konfigurationsdatei außer Kraft.

Verwenden Sie dann den folgenden Code, um den Proxy für die Verwendung bestimmter Anmeldeinformationen zu konfigurieren:

```
IDictionary channelProperties =
                        ChannelServices.GetChannelSinkProperties(proxy);
NetworkCredential credentials;
credentials = new NetworkCredential("username", "password", "domain");
ObjRef objectReference = RemotingServices.Marshal(proxy);
Uri objectUri = new Uri(objectReference.URI);
CredentialCache credCache = new CredentialCache();
// Substitute "authenticationType" with "Negotiate", "Basic", "Digest",
// "Kerberos" or "NTLM"
credCache.Add(objectUri, "authenticationType", credentials);
channelProperties["credentials"] = credCache;
channelProperties["preauthenticate"] = true;
```

Immer einen bestimmten Authentifizierungstyp anfordern

Sie sollten immer einen bestimmten Authentifizierungstyp unter Verwendung der Methode **CredentialCache.Add** anfordern, wie oben gezeigt wurde. Vermeiden Sie die direkte Verwendung der Klasse **NetworkCredential**, wie im nachfolgenden Code veranschaulicht wird:

```
IDictionary providerData = ChannelServices.GetChannelSinkProperties(yourProxy);
providerData["credentials"] = new NetworkCredential(uid, pwd);
```

Dies sollte im Produktionscode vermieden werden, da Sie keine Kontrolle über den vom Remoteobjekthost verwendeten Authentifizierungsmechanismus haben. Somit haben Sie auch keine Kontrolle über die verwendeten Anmeldeinformationen.

Es könnte z. B. vorkommen, dass Sie eine Kerberos- oder NTLM-Authentifizierungsherausforderung vom Server erwarten, jedoch stattdessen eine Standardherausforderung erhalten. In diesem Fall werden der bereitgestellte Benutzername und das Kennwort unverschlüsselt in Textform an den Server gesendet.

Festlegen der Eigenschaft "PreAuthenticate"

Die Eigenschaft **PreAuthenticate** des Proxys kann auf den Wert **True** (Wahr) oder **False** (Falsch) gesetzt werden. Legen Sie den Wert auf **True** fest (wie im o. a. Code dargestellt), um bestimmte Authentifizierungsanmeldeinformationen bereitzustellen, die dazu führen, dass mit der ersten Anforderung der HTTP-Header **WWW-Authenticate** übergeben wird. Dadurch muss der Webserver weder den Zugriff für die erste Anforderung ablehnen noch die Authentifizierung für die nachfolgende Anforderung durchführen.

Verwenden der Eigenschaft "ConnectionGroupName"

Wenn Sie eine ASP.NET-Webanwendung besitzen, die eine Verbindung zu einer Remotekomponente (von ASP.NET verwaltet) herstellt und den Sicherheitskontext des ursprünglichen Aufrufers übermittelt (durch Verwendung von **DefaultCredentials** und des Identitätswechsels oder durch Festlegen von expliziten Anmeldeinformationen, wie zuvor gezeigt), sollten Sie die Eigenschaft **ConnectionGroupName** des Kanals innerhalb der Webanwendung festlegen. Dadurch wird verhindert, dass ein neuer, nicht authentifizierter Client eine bestehende, authentifizierte TCP-Verbindung zur Remotekomponente wiederverwenden kann, die den Authentifizierungsanmeldeinformationen eines vorherigen Clients zugeordnet ist. Die Wiederverwendung von Verbindungen kann als Ergebnis von **HTTP KeepAlives** und der Authentifizierungspersistenz auftreten, die in IIS aus Leistungsgründen aktiviert ist.

Legen Sie für die Eigenschaft **ConnectionGroupName** einen Bezeichner fest (z. B. den Benutzernamen des Aufrufers), der einen Aufrufer vom nächsten unterscheidet.

```
channelProperties["connectiongroupname"] = userName;
```

Hinweis: Wenn der Sicherheitskontext des ursprünglichen Aufrufers nicht über die Webanwendung an die Remotekomponente übermittelt wird, sondern die Verbindung zur Remotekomponente mithilfe einer festen Identität (z. B. die ASP.NET-Prozessidentität der Webanwendung) herstellt, müssen Sie die Eigenschaft **ConnectionGroupName** nicht festlegen. In diesem Szenario bleibt der Sicherheitskontext der Verbindung beim Wechsel von einem Aufrufer zum nächsten konstant.

Die nächste Version des .NET Frameworks unterstützt das Verbindungspooling auf Basis der SID des Threads, der das Proxyobjekt aufruft. Dadurch wird das oben beschriebene Problem behandelt, wenn die Webanwendung für den Aufrufer einen Identitätswechsel durchführt. Das Pooling wird für .NET Remoting-Clients unterstützt, jedoch nicht für Webdienstclients.

Übermitteln des ursprünglichen Aufrufers

In diesem Abschnitt wird beschrieben, wie Sie den Sicherheitskontext des ursprünglichen Aufrufers über eine ASP.NET-Webanwendung an eine Remotekomponente übermitteln können, die von ASP.NET auf einem Remoteanwendungsserver verwaltet wird. Sie müssen dies möglicherweise durchführen, um die Autorisierung auf Benutzerbasis innerhalb des Remoteobjekts oder innerhalb nachgeschalteter Subsysteme (z. B. Datenbanken) zu unterstützen.

In Abbildung 11.4 wird der Sicherheitskontext des ursprünglichen Aufrufers (Bob) über den Front-End-Webserver, der als Host für eine ASP.NET-Webanwendung fungiert, an das Remoteobjekt übermittelt, das von ASP.NET auf einem Remoteanwendungsserver verwaltet und abschließend an einen Back-End-Datenbankserver übermittelt wird.

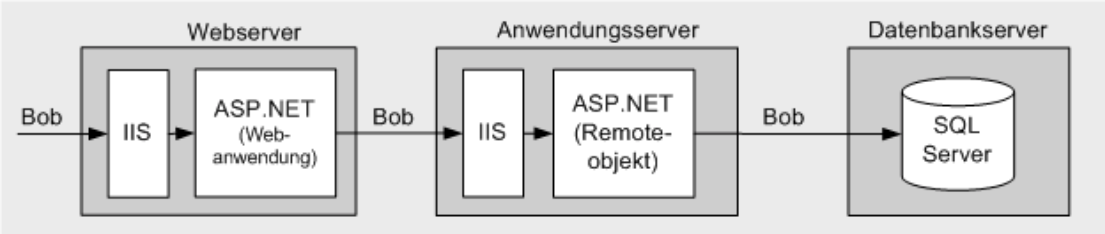

Abbildung 11.4
Übermitteln des Sicherheitskontextes des ursprünglichen Aufrufers

Damit die Anmeldeinformationen an ein Remoteobjekt übermittelt werden, muss der Remoteobjektclient (in diesem Szenario die ASP.NET-Webanwendung) den Proxy des Objekts konfigurieren und die Eigenschaft **Credentials** des Proxys explizit festlegen, wie weiter oben in diesem Kapitel unter "Übergeben von Anmeldeinformationen an Remoteobjekte zur Authentifizierung" beschrieben.

Hinweis: IPrincipal-Objekte werden nicht über .NET Remoting-Grenzen hinweg übermittelt.

Es bieten sich zwei Möglichkeiten, um den Kontext des Aufrufers zu übermitteln:

- **Übergeben von Standardanmeldeinformationen und Verwenden der Kerberos-Authentifizierung (und -Delegierung) -** Dieser Ansatz erfordert, dass Sie innerhalb der ASP.NET-Webanwendung einen Identitätswechsel durchführen und den Remoteobjektproxy mit **DefaultCredentials** konfigurieren, das vom Sicherheitskontext des Aufrufers nach dem Identitätswechsel abgerufen wird.
- **Übergeben expliziter Anmeldeinformationen und Verwenden der Standard- oder Formularauthentifizierung -** Dieser Ansatz erfordert keinen Identitätswechsel innerhalb der ASP.NET-Webanwendung. Stattdessen konfigurieren Sie den Remoteobjektproxy programmatisch mit expliziten Anmeldeinformationen, die entweder von Servervariablen (über die Standardauthentifizierung) oder von HTML-Formularfeldern (über die Formularauthentifizierung) abgerufen werden, die für die Webanwendung zur Verfügung stehen. Bei der Standard- oder Formularauthentifizierung stehen der Benutzername und das Kennwort unverschlüsselt für den Server zur Verfügung.

Standardanmeldeinformationen mit Kerberos-Delegierung

Alle Computer (Server und Clients) müssen Windows 2000 oder eine nachfolgende Version ausführen, um die Kerberos-Delegierung verwenden zu können. Zusätzlich müssen Clientkonten, die delegiert werden sollen, im Verzeichnisdienst Active Directory™ gespeichert werden und dürfen nicht als **Konto ist vertraulich und kann nicht delegiert werden** gekennzeichnet werden.

In den nachfolgenden Tabellen werden die Konfigurationsschritte aufgeführt, die auf dem Webserver und Anwendungsserver erforderlich sind.

Konfigurieren des Webservers

Konfigurieren von IIS	
Schritt	**Weitere Informationen**
Deaktivieren des anonymen Zugriffs für das virtuelle Stammverzeichnis der Webanwendung Aktivieren der integrierten Windows-Authentifizierung für das virtuelle Stammverzeichnis der Webanwendung	Die Kerberos-Authentifizierung wird unter der Annahme verhandelt, dass die Clients und Server Windows 2000 oder höher ausführen. **Hinweis**: Wenn Sie Microsoft Internet Explorer 6 unter Windows 2000 verwenden, wird standardmäßig die NTLM-Authentifizierung anstelle der erforderlichen Kerberos-Authentifizierung verwendet. Informationen zum Aktivieren der Kerberos-Delegierung finden Sie in der Microsoft Knowledge Base im Artikel Q299838, "Unable to Negotiate Kerberos Authentication after upgrading to Internet Explorer 6" (US).
Konfigurieren von ASP.NET	
Schritt	**Weitere Informationen**
Konfigurieren der ASP.NET-Webanwendung für die Windows-Authentifizierung	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Webanwendung. Legen Sie das <**authentication**>-Element fest auf: `<authentication mode="Windows" />`
Konfigurieren der ASP.NET-Webanwendung für Identitätswechsel	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Webanwendung. Legen Sie das <**identity**>-Element fest auf: `<identity impersonate="true" />`
Konfigurieren des Remoting (clientseitiger Proxy)	
Schritt	**Weitere Informationen**

Konfigurieren des Remoteobjektproxys für die Verwendung von Standardanmeldeinformationen für alle Aufrufe des Remoteobjekts	Fügen Sie folgenden Eintrag zur Datei **Web.config** hinzu:
	
	`<channel ref="http"`
	`useDefaultCredentials="true" />`
	
	Die Anmeldeinformationen werden vom Identitätswechseltoken des Threads der Webanwendung abgerufen.

Konfigurieren des Remoteanwendungsservers

Konfigurieren von IIS	
Schritt	Weitere Informationen
Deaktivieren des anonymen Zugriffs für das virtuelle Stammverzeichnis der Webanwendung Aktivieren der integrierten Windows-Authentifizierung für das virtuelle Stammverzeichnis der Webanwendung	
Konfigurieren von ASP.NET (Remoteobjekthost)	
Schritt	Weitere Informationen
Konfigurieren von ASP.NET, sodass die Windows-Authentifizierung verwendet wird	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Anwendung. Legen Sie das <**authentication**>-Element fest auf: `<authentication mode="Windows" />`
Konfigurieren Sie ASP.NET für den Identitätswechsel	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Anwendung. Legen Sie das <**identity**>-Element fest auf: `<identity impersonate="true" />` **Hinweis**: Dieser Schritt ist nur erforderlich, wenn der Sicherheitskontext des ursprünglichen Aufrufers über das Remoteobjekt an das nächste nachgeschaltete Subsystem (z. B. eine Datenbank) übermittelt werden soll. Wenn hier der Identitätswechsel aktiviert ist, wird beim Ressourcenzugriff (lokal und remote) der Sicherheitskontext des ursprünglichen Aufrufers nach dem Identitätswechsel verwendet. Wenn Sie einfach Autorisierungsüberprüfungen im Remoteobjekt auf Benutzerbasis ermöglichen möchten, müssen Sie hier keinen Identitätswechsel durchführen.

Weitere Informationen

Weitere Informationen zur Kerberos-Delegierung finden Sie unter "Vorgehensweise: Implementieren der Kerberos-Delegierung unter Windows 2000" im Abschnitt "Referenz" dieses Handbuchs.

Explizite Anmeldeinformationen bei der Standard- oder Formularauthentifizierung

Sie können als Alternative zur Kerberos-Delegierung die Standard- oder Formularauthentifizierung für die Webanwendung verwenden, um die Anmeldeinformationen des Clients zu erfassen und dann die Standardauthentifizierung (oder integrierte Windows-Authentifizierung) für das Remoteobjekt verwenden.

Bei diesem Ansatz stehen die unverschlüsselten Anmeldeinformationen des Clients für die Webanwendung zur Verfügung. Diese können über den Remoteobjektproxy an ein Remoteobjekt übergeben werden. Dazu müssen Sie Code in die Webanwendung einbeziehen, um die Anmeldeinformationen des Clients abzurufen und den Remoteobjektproxy zu konfigurieren.

Standardauthentifizierung

Bei der Standardauthentifizierung stehen die Anmeldeinformationen des ursprünglichen Aufrufers für die Webanwendung in Servervariablen zur Verfügung. Der folgende Code zeigt, wie diese abgerufen werden und wie das Konfigurieren des Remoteobjektproxys erfolgt.

```
// Retrieve client's credentials (available with Basic authentication)
string pwd = Request.ServerVariables["AUTH_PASSWORD"];
string uid = Request.ServerVariables["AUTH_USER"];
// Associate the credentials with the remote object proxy
IDictionary channelProperties =
                        ChannelServices.GetChannelSinkProperties(proxy);
NetworkCredential credentials;
credentials = new NetworkCredential(uid, pwd);
ObjRef objectReference = RemotingServices.Marshal(proxy);
Uri objectUri = new Uri(objectReference.URI);
CredentialCache credCache = new CredentialCache();
credCache.Add(objectUri, "Basic", credentials);
channelProperties["credentials"] = credCache;
channelProperties["preauthenticate"] = true;
```

Hinweis: Der im o. a. Code gezeigte **NetworkCredential**-Konstruktor wird mit der Benutzer-ID und dem Kennwort bereitgestellt. Es kann eine Standarddomäne auf dem Webserver in IIS beim Konfigurieren der Standardauthentifizierung konfiguriert werden, um die feste Kodierung des Domänennamens zu vermeiden.

Formularauthentifizierung

Bei der Formularauthentifizierung stehen die Anmeldeinformationen des ursprünglichen Aufrufers für die Webanwendung in Formularfeldern (anstatt in Servervariablen) zur Verfügung. In diesem Fall verwenden Sie den folgenden Code:

```
// Retrieve client's credentials from the logon form
string pwd = txtPassword.Text;
string uid = txtUid.Text;
// Associate the credentials with the remote object proxy
IDictionary channelProperties =
                        ChannelServices.GetChannelSinkProperties(proxy);
NetworkCredential credentials;
```

```
credentials = new NetworkCredential(uid, pwd);
ObjRef objectReference = RemotingServices.Marshal(proxy);
Uri objectUri = new Uri(objectReference.URI);
CredentialCache credCache = new CredentialCache();
credCache.Add(objectUri, "Basic", credentials);
channelProperties["credentials"] = credCache;
channelProperties["preauthenticate"] = true;
```

In den nachfolgenden Tabellen werden die Konfigurationsschritte aufgeführt, die auf dem Webserver und Anwendungsserver erforderlich sind.

Konfigurieren des Webservers

Konfigurieren von IIS	
Schritt	Weitere Informationen
Deaktivieren des anonymen Zugriffs für das virtuelle Stammverzeichnis der Webanwendung und Auswählen der Standardauthentifizierung, wenn diese verwendet werden soll – oder – Aktivieren des anonymen Zugriffs, wenn die Formularauthentifizierung verwendet werden soll	Sowohl die Standard- als auch die Formularauthentifizierung sollten gemeinsam mit SSL verwendet werden, um die unverschlüsselten Anmeldeinformationen zu schützen, die über das Netzwerk gesendet werden. Wenn Sie die Standardauthentifizierung verwenden, sollten Sie in allen Seiten (nicht nur auf der ersten Anmeldeseite) SSL verwenden, da die Standardanmeldeinformationen bei jeder Anforderung übergeben werden. Ebenso sollte SSL für alle Seiten verwendet werden, wenn Sie die Formularauthentifizierung verwenden, um die unverschlüsselten Anmeldeinformationen bei der ersten Anmeldung sowie das Authentifizierungsticket zu schützen, das bei nachfolgenden Anforderungen übergeben wird.
Konfigurieren von ASP.NET	
Schritt	Weitere Informationen

Konfigurieren der ASP.NET-Webanwendung für die Verwendung der Windows-Authentifizierung, wenn die Standardauthentifizierung verwendet wird
 	
– oder –
 	
Konfigurieren der ASP.NET-Webanwendung für die Verwendung der Formularauthentifizierung, wenn die Formularauthentifizierung verwendet wird
	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Webanwendung.
	Legen Sie das <**authentication**>-Element fest auf:
	
	`<authentication mode="Windows" />`
	
	– oder –
	
	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Webanwendung.
	Legen Sie das <**authentication**>-Element fest auf:
	
	`<authentication mode="Forms" />`

Deaktivieren des Identitätswechsels innerhalb der ASP.NET-Webanwendung	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Webanwendung. Legen Sie das <**identity**>-Element fest auf: `<identity impersonate="false" />` **Hinweis**: Dies ist damit vergleichbar, dass kein <**identity**>-Element vorhanden ist. Der Identitätswechsel ist nicht erforderlich, da die Anmeldeinformationen des Clients explizit über den Remoteobjektproxy an das Remoteobjekt übergeben werden.
Konfigurieren des Remoting (clientseitiger Proxy)	
Schritt	Weitere Informationen

Konfigurieren des Remoting-Proxys, damit dieser nicht die Standardanmeldeinformationen für alle Aufrufe des Remoteobjekts verwendet	Fügen Sie folgenden Eintrag zur Datei **Web.config** hinzu:
	
	`<channel ref="http"`
	`useDefaultCredentials="false" />`
	
	Sie möchten nicht, dass Standardanmeldeinformationen verwendet werden (da die Webanwendung so konfiguriert ist, dass sie keinen Identitätswechsel durchführt. Dies würde dazu führen, dass der Sicherheitskontext der ASP.NET-Prozessidentität verwendet wird).

Schreiben von Code, um die Anmeldeinformationen für den Remoteobjektproxy zu erfassen und explizit festzulegen	Siehe hierzu die zuvor gezeigten Codefragmente.

Konfigurieren des Anwendungsservers

Konfigurieren von IIS	
Schritt	Weitere Informationen
Deaktivieren des anonymen Zugriffs für das virtuelle Stammverzeichnis der Anwendung	
Aktivieren der Standardauthentifizierung	**Hinweis:** Die Standardauthentifizierung auf dem Anwendungsserver (Remoteobjekt) ermöglicht es dem Remoteobjekt, den Sicherheitskontext des ursprünglichen Aufrufers an die Datenbank zu übermitteln (da der Benutzername und das Kennwort des Aufrufers unverschlüsselt zur Verfügung stehen und zum Antworten auf Herausforderungen des Datenbankservers für Netzwerkauthentifizierungen verwendet werden können). Wenn der Sicherheitskontext des ursprünglichen Aufrufers nicht über das Remoteobjekt hinaus übermittelt werden muss, erwägen Sie die Konfiguration von IIS auf dem Anwendungsserver für die Verwendung der integrierten Windows-Authentifizierung, da diese eine umfassendere Sicherheit bietet, d. h. Anmeldeinformationen werden nicht über das Netzwerk gesendet und stehen der Anwendung nicht zur Verfügung.
Konfigurieren von ASP.NET (Remoteobjekthost)	
Schritt	Weitere Informationen
Konfigurieren von ASP.NET, sodass die Windows-Authentifizierung verwendet wird	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Anwendung. Legen Sie das <**authentication**>-Element fest auf: `<authentication mode="Windows" />`
Konfigurieren von ASP.NET für den Identitätswechsel	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Anwendung. Legen Sie das <**identity**>-Element fest auf: `<identity impersonate="true" />` **Hinweis**: Dieser Schritt ist nur erforderlich, wenn der Sicherheitskontext des ursprünglichen Aufrufers über das Remoteobjekt an das nächste nachgeschaltete Subsystem (z. B. eine Datenbank) übermittelt werden soll. Wenn hier der Identitätswechsel aktiviert ist, wird beim Ressourcenzugriff (lokal und remote) der Sicherheitskontext des ursprünglichen Aufrufers nach dem Identitätswechsel verwendet. Wenn Sie einfach Autorisierungsüberprüfungen im Remoteobjekt auf Benutzerbasis ermöglichen möchten, müssen Sie hier keinen Identitätswechsel durchführen.

Vertrauenswürdiges Subsystem

Das Modell des vertrauenswürdigen Subsystems bietet einen alternativen (und einfach zu implementierenden) Ansatz zum Übermitteln des Sicherheitskontextes des ursprünglichen Aufrufers. In diesem Modell besteht zwischen dem Remoteobjekthost und der Webanwendung eine Vertrauensgrenze. Das Remoteobjekt vertraut der Webanwendung, um die Aufrufer ordnungsgemäß zu authentifizieren und zu autorisieren, bevor die Anforderungen an das Remoteobjekt weitergeleitet werden. Im Remoteobjekthost wird keine Authentifizierung des ursprünglichen Aufrufers durchgeführt. Der Remoteobjekthost authentifiziert die feste vertrauenswürdige Identität, die von der Webanwendung zum Kommunizieren mit dem Remoteobjekt verwendet wird. In den meisten Fällen ist dies die Prozessidentität der ASP.NET-Webanwendung.

Das Modell mit vertrauenswürdigen Subsystemen ist in Abbildung 11.5 dargestellt. Dieses Diagramm zeigt ebenfalls zwei mögliche Konfigurationen. Die erste Konfiguration verwendet den ASP.NET-Host und den HTTP-Kanal, während die zweite Konfiguration einen Windows-Diensthost und den TCP-Kanal verwendet.

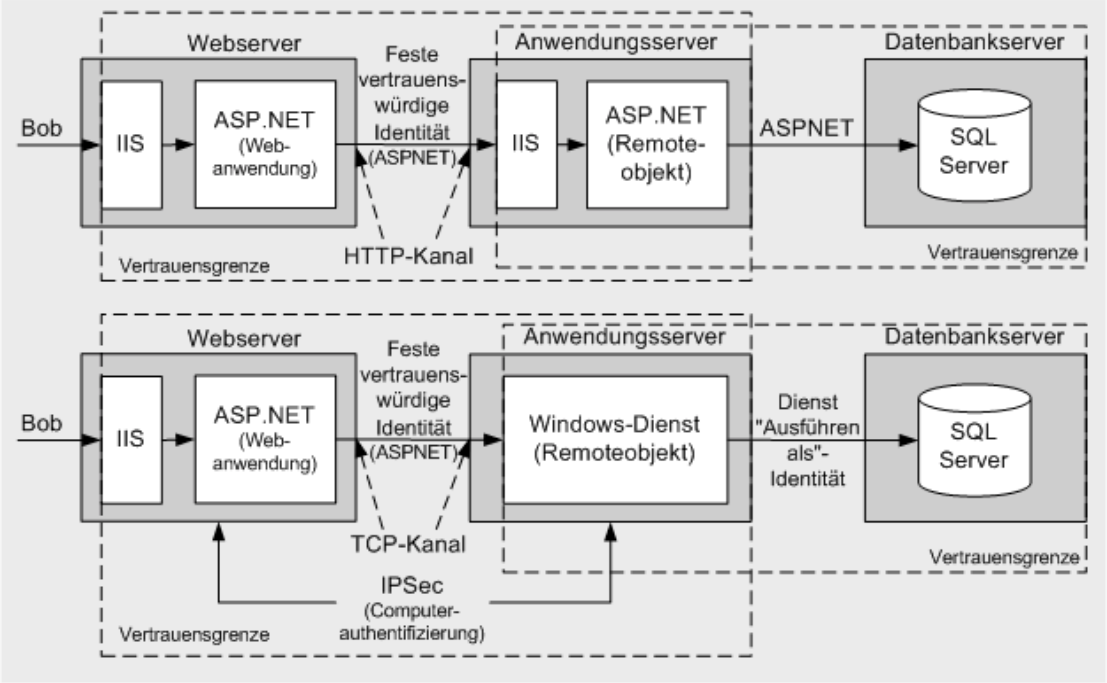

Abbildung 11.5
Das Modell mit vertrauenswürdigen Subsystemen

Übermitteln der Aufruferidentität

Wenn Sie das vertraute Subsystemmodell verwenden, müssen Sie möglicherweise dennoch die Identität des ursprünglichen Aufrufers (Name, nicht Sicherheitskontext) übermitteln, z. B. für Überwachungsaufgaben bei der Datenbank.

Sie können die Identität auf Anwendungsebene übermitteln, indem Sie die Parameter von Methoden und gespeicherten Prozeduren verwenden, während Parameter von vertrauten Abfragen (wie im folgenden Beispiel gezeigt) zum Abrufen benutzerspezifischer Daten aus der Datenbank verwendet werden können.

```
SELECT x,y,z FROM SomeTable WHERE UserName = "Bob"
```

Auswählen eines Hosts

Vertrautes Subsystemmodell bedeutet, dass der Remoteobjekthost die ursprünglichen Aufrufer nicht authentifiziert. Er muss jedoch weiterhin seinen unmittelbaren Client (in diesem Szenario die ASP.NET-Webanwendung) authentifizieren (und autorisieren), damit keine nicht autorisierten Anwendungen Anforderungen an Remoteobjekte senden.

Wenn das Hosting in ASP.NET erfolgt und Sie den HTTP-Kanal verwenden, können Sie die integrierte Windows-Authentifizierung verwenden, um die Prozessidentität der ASP.NET-Webanwendung zu authentifizieren.

Wenn das Hosting in einem Windows-Dienst erfolgt, können Sie den TCP-Kanal verwenden, der hervorragende Leistungen, aber keine Authentifizierungsmöglichkeiten bietet. In diesem Szenario können Sie zwischen dem Webserver und dem Anwendungsserver IPSec verwenden. Es kann eine IPSec-Richtlinie eingerichtet werden, die es nur dem Webserver ermöglicht, mit dem Anwendungsserver zu kommunizieren.

Konfigurationsschritte

In den nachfolgenden Tabellen werden die Konfigurationsschritte aufgeführt, die auf dem Webserver und Anwendungsserver erforderlich sind.

Konfigurieren des Webservers

Konfigurieren von IIS	
Schritt	Weitere Informationen
Konfigurieren der IIS-Authentifizierung	Die Webanwendung kann eine beliebige Form der Authentifizierung verwenden, um die ursprünglichen Aufrufer zu authentifizieren.
Konfigurieren von ASP.NET	
Schritt	Weitere Informationen

Konfigurieren der Authentifizierung und Sicherstellen, dass der Identitätswechsel deaktiviert ist	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Webanwendung.
	Legen Sie das <**authentication**>-Element auf **Windows** (Windows-Authentifizierung), **Forms** (Formularauthentifizierung) oder **Passport** (Passport-Authentifizierung) fest.
	
	`<authentication mode="Windows|Forms|Passport" />`
	
	Legen Sie das <**identity**>-Element fest auf:
	
	`<identity impersonate="false" />`
	
	(oder entfernen Sie das Element <**identity**>)

Zurücksetzen des Kennworts des ASPNET-Kontos, das zum Ausführen von ASP.NET verwendet wird – oder – Erstellen eines Domänenkontos mit minimalen Rechten zum Ausführen von ASP.NET und Festlegen von Kontodetails für das Element <**processModel**> in der Datei **Web.config**	Weitere Informationen zum Zugreifen auf Netzwerkressourcen (einschließlich der Remoteobjekte) von ASP.NET und zum Auswählen und Konfigurieren eines Prozesskontos für ASP.NET finden Sie unter "Zugreifen auf Netzwerkressourcen" und "Prozessidentität für ASP.NET" in Kapitel 8, "ASP.NET-Sicherheit".
Konfigurieren des Remoting (clientseitiger Proxy)	
Schritt	Weitere Informationen

Konfigurieren des Remoting-Proxys, damit dieser die Standardanmeldeinformationen für alle Aufrufe des Remoteobjekts verwendet	Fügen Sie folgenden Eintrag zur Datei **Web.config** hinzu:
	
	`<channel ref="http"`
	`useDefaultCredentials="true" />`
	
	Da die Webanwendung keinen Identitätswechsel durchführt, führt die Verwendung von Standardanmeldeinformationen zum Verwenden der ASP.NET-Prozessidentität für alle Aufrufe des Remoteobjekts.

Konfigurieren des Anwendungsservers

Wenn Sie einen ASP.NET-Host verwenden, gelten die folgenden Schritte.

Konfigurieren von IIS	
Schritt	Weitere Informationen
Deaktivieren des anonymen Zugriffs für das virtuelle Stammverzeichnis der Anwendung Aktivieren der integrierten Windows-Authentifizierung	
Konfigurieren von ASP.NET (Remoteobjekthost)	
Schritt	Weitere Informationen
Konfigurieren von ASP.NET, sodass die Windows-Authentifizierung verwendet wird	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Anwendung. Legen Sie das <**authentication**>-Element fest auf: `<authentication mode="Windows" />`
Deaktivieren des Identitätswechsels	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis der Anwendung. Legen Sie das <**identity**>-Element fest auf: `<identity impersonate="false" />`

Verwenden eines Windows-Diensthosts

Wenn Sie einen Windows-Diensthostprozess verwenden, müssen Sie ein Windows-Konto zum Ausführen des Dienstes erstellen. Dieser Sicherheitskontext, der von diesem Konto bereitgestellt wird, wird vom Remoteobjekt für alle lokalen und Remoteressourcenzugriffe verwendet.

Für den Zugriff auf eine Microsoft SQL Server™-Remotedatenbank (unter Verwendung der Windows-Authentifizierung) können Sie ein Domänenkonto oder lokales Konto mit minimalen Rechten verwenden und dann ein dupliziertes Konto (mit demselben Benutzernamen und demselben Kennwort) auf dem Datenbankserver erstellen.

Sichere Kommunikation

Die sichere Kommunikation befasst sich mit der garantierten Integrität und Vertraulichkeit von Nachrichten, während diese über das Netzwerk übermittelt werden. Sie können einen plattformbasierten Ansatz verwenden, um die Kommunikation zu sichern und SSL oder IPSec zu verwenden, oder Sie verwenden einen Ansatz auf Nachrichtenebene und entwickeln eine benutzerdefinierte Verschlüsselungssenke, um die gesamte Nachricht oder ausgewählte Teile einer Nachricht zu verschlüsseln.

Plattformebenenoptionen

Die beiden Optionen der Plattformebene, die zum Sichern der zwischen einem Client und einer Remotekomponente zu übertragenden Daten in Betracht gezogen werden, sind Folgende:

- SSL
- IPSec

Wenn Sie Remoteobjekte in ASP.NET verwalten, können Sie SSL zum Sichern des Kommunikationskanals zwischen Client und Server verwenden. Dies erfordert ein Serverauthentifizierungszertifikat auf dem Computer, der das Remoteobjekt verwaltet.

Wenn Sie Remoteobjekte in einem Windows-Dienst verwalten, können Sie IPSec zwischen den Client- und Hostcomputern (Server) verwenden oder eine benutzerdefinierte Verschlüsselungssenke entwickeln.

Nachrichtenebenenoptionen

Aufgrund der erweiterbaren Natur der .NET Remoting-Architektur können Sie eigene benutzerdefinierte Senken entwickeln und diese in die Verarbeitungspipeline einfügen. Sie können eine benutzerdefinierte Senke entwickeln, die die Nachrichtendaten verschlüsselt und entschlüsselt, die vom oder an das Remoteobjekt gesendet werden, um die sichere Kommunikation bereitzustellen.

Der Vorteil dieses Ansatzes ist, dass er es Ihnen ermöglicht, ausgewählte Teile einer Nachricht zu verschlüsseln. Dies steht im Gegensatz zu den Ansätzen auf Plattformebene, die alle Daten verschlüsseln, die zwischen Client und Server gesendet werden.

Weitere Informationen

Weitere Informationen zu SSL und IPSec finden Sie in Kapitel 4, "Sichere Kommunikation".

Auswählen eines Hostprozesses

Objekte, auf die der Remotezugriff erfolgen soll, müssen in einer ausführbaren Hostdatei ausgeführt werden. Der Host achtet auf eingehende Anforderungen und sendet Aufrufe an Objekte. Der ausgewählte Hosttyp wirkt sich auf den Nachrichtentransportmechanismus aus, der als Kanal bezeichnet wird. Der von Ihnen ausgewählte Kanal wirkt sich auf die Authentifizierung, die Autorisierung, die sichere Kommunikation sowie auf die Leistungsmerkmale Ihrer Lösung aus.

Der HTTP-Kanal bietet bessere Sicherheitsoptionen, während der TCP-Kanal die bessere Leistung aufweist.

Ihnen stehen die folgenden Hauptoptionen für das Hosting von Remoteobjekten zur Verfügung:

- Hosting in ASP.NET
- Hosting in einem Windows-Dienst
- Hosting in einer Konsolenanwendung

Empfehlung

Damit Sie den Vorteil der von ASP.NET und IIS bereitgestellten Sicherheitsinfrastruktur nutzen können, wird hinsichtlich der Sicherheit empfohlen, das Hosting für Remoteobjekte in ASP.NET durchzuführen. Dies erfordert, dass die Clients mit den Remoteobjekten über den HTTP-Kanal kommunizieren. Die ASP.NET- und IIS-Funktionen für die Authentifizierung, Autorisierung und die sichere Kommunikation stehen für Remoteobjekte zur Verfügung, die in ASP.NET verwaltet werden.

Erwägen Sie das Hosting von Remoteobjekten in Windows-Diensten, wenn die Leistung Ihr Hauptanliegen darstellt (und nicht die Sicherheit).

Hosting in ASP.NET

Folgendes trifft zu, wenn Sie ein Remoteobjekt in ASP.NET verwalten:

- Auf das Objekt wird über das HTTP-Protokoll zugegriffen.
- Das Objekt besitzt einen Endpunkt, der über einen URL zur Verfügung steht.
- Es existiert in einer Anwendungsdomäne innerhalb des Workerprozesses **Aspnet_wp.exe**.
- Es erbt die von IIS und ASP.NET angebotenen Sicherheitsfunktionen.

Vorteile

Wenn Sie Remoteobjekte in IIS verwalten, bieten sich Ihnen folgende Vorteile:

- Von IIS und ASP.NET bereitgestellte Funktionen für die Authentifizierung, Autorisierung und sichere Kommunikation stehen unmittelbar zur Verfügung.
- Sie können die Überwachungsfunktionen von IIS verwenden.
- Der ASP.NET-Workerprozess ist immer aktiv.
- Sie besitzen umfangreiche Steuerungsmöglichkeiten für die ausführbare Hostdatei über das Element <**processModel**> in der Datei **Machine.config**. Sie können die Threadverwaltung, die Fehlertoleranz, die Speicherverwaltung usw. steuern.
- Sie können vor dem Remoteobjekt eine Nachbildungsebene für einen Webdienst erstellen.

Nachteile

Wenn Sie ASP.NET für das Hosting von Remoteobjekten verwenden, sollten Sie sich der folgenden Nachteile bewusst sein:

- Es erfordert die Verwendung des HTTP-Kanals, der langsamer als der TCP-Kanal ist.
- Benutzerprofile werden nicht von ASP.NET geladen. Verschiedene Verschlüsselungsverfahren (einschließlich DPAPI) erfordern möglicherweise Benutzerprofile.
- Wenn Code auf das Objekt zugreift, der in einer ASP.NET-Webanwendung ausgeführt wird, müssen Sie möglicherweise die Standardauthentifizierung verwenden.

Hosting in einem Windows-Dienst

Wenn Sie ein Remoteobjekt in einem Windows-Dienst verwalten, existiert das Remoteobjekt in einer Anwendungsdomäne, die im Dienstprozess enthalten ist. Sie können nicht den HTTP-Kanal verwenden, sondern Sie müssen den TCP-Kanal verwenden. Der TCP-Kanal unterstützt die folgenden Sicherheitsfunktionen:

- **Authentifizierungsfunktionen**
  Sie müssen eine benutzerdefinierte Authentifizierungslösung bereitstellen. Die Optionen umfassen Folgendes:
  - **Verwenden der zugrunde liegenden Authentifizierungsdienste der SSPI -** Sie können eine Kanalsenke erstellen, die die Anmeldeinformationen der Windows-SSPI und Kontextverwaltungs-APIs verwendet, um den Aufrufer zu authentifizieren und optional einen Identitätswechsel für diesen durchzuführen. Die Kanalsenke ist oberhalb des TCP-Kanals angesiedelt. Die SSPI ermöglicht es dem Client und Server in Verbindung mit dem TCP-Kanal, Authentifizierungsinformationen auszutauschen. Nach der Authentifizierung können der Client und der Server Nachrichten senden, die die Integrität und Vertraulichkeit sicherstellen.
  - **Verwenden eines zugrunde liegenden Transports, der die Authentifizierung unterstützt, z. B. Named Pipes -** Der Named Pipe-Kanal verwendet Named Pipes als Transportmechanismus. Dadurch wird die Authentifizierung für den Aufrufer bereitgestellt, und es wird die Sicherheit auf Basis der Windows-Zugriffssteuerungslisten für die Pipe eingeführt und der Identitätswechsel für den Aufrufer durchgeführt.
- **Autorisierungsfunktionen**
  Die Autorisierung ist nur möglich, wenn Sie eine benutzerdefinierte Authentifizierungslösung implementieren.
  - Wenn Sie in der Lage sind, den Identitätswechsel für den Benutzer durchzuführen (z. B. durch Verwenden eines zugrunde liegenden Named Pipe-Transports), können Sie **WindowsPrincipal.IsInRole** verwenden.
  - Wenn Sie ein **IPrincipal**-Objekt erstellen können, um den authentifizierten Client darzustellen, können Sie .NET-Rollen verwenden (über PrincipalPermission-Forderungen und explizite Rollenüberprüfungen mithilfe von **IPrincipal.IsInRole**).
- **Sichere Kommunikationsfunktionen**
  Ihnen bieten sich zwei Optionen:
  - Verwenden Sie IPSec, um den Transport von Daten zwischen dem Client und dem Server zu sichern.
  - Erstellen Sie eine benutzerdefinierte Kanalsenke, die eine asymmetrische Verschlüsselung durchführt. Diese Option wird weiter unten in diesem Kapitel erläutert.

Vorteile

Windows-Dienste als Host für Remoteobjekte zu verwenden, hat folgende Vorteile:

- Ein hoher Grad an Steuerungsmöglichkeiten für die Aktivierung des Hostprozesses
- Erbt die Vorteile der Windows-Dienstarchitektur
- Keine Notwendigkeit, IIS auf der mittleren Ebene Ihrer Anwendung einzuführen
- Benutzerprofile werden automatisch geladen
- Die Leistung ist gut, da die Clients mithilfe binär verschlüsselter Daten über den TCP-Kanal kommunizieren

Nachteile

Wenn Sie einen Windows-Dienst als Host für Remoteobjekte verwenden, sollten Sie sich der folgenden Nachteile bewusst sein:

- Sie müssen benutzerdefinierte Authentifizierungs- und Autorisierungslösungen bereitstellen.
- Sie müssen sichere Kommunikationslösungen bereitstellen.
- Sie müssen Überwachungslösungen bereitstellen.

Hosting in einer Konsolenanwendung

Wenn Sie ein Remoteobjekt in einer Konsolenanwendung verwalten, existiert das Remoteobjekt in einer Anwendungsdomäne, die im Konsolenanwendungsprozess enthalten ist. Sie können nicht den HTTP-Kanal verwenden, sondern Sie müssen den TCP-Kanal verwenden.

Dieser Ansatz wird für Produktionslösungen nicht empfohlen.

Vorteile

Für diesen Ansatz gibt es wenige Vorteile, obwohl es bedeutet, dass IIS nicht auf der mittleren Ebene erforderlich ist. Dieser Ansatz wird jedoch nur für die Entwicklung und zum Testen empfohlen und nicht für Produktionsumgebungen.

Nachteile

Wenn Sie eine benutzerdefinierte ausführbare Datei für das Hosting von Remoteobjekten verwenden, sollten Sie sich der folgenden Nachteile bewusst sein:

- Der Host muss manuell gestartet und unter der interaktiven Anmeldesitzung ausgeführt werden (nicht empfehlenswert).
- Es ist keine Fehlertoleranz vorhanden.
- Sie müssen eine benutzerdefinierte Authentifizierung und Autorisierung bereitstellen.
- Es sind keine Überwachungsfunktionen vorhanden.

Remoting und Webdienste

.NET bietet viele verschiedene Verfahren, um es Clients zu ermöglichen, mit Remoteobjekten (einschließlich der Verwendung von Webdiensten) zu kommunizieren.

Wenn die Interoperabilität zwischen heterogenen Systemen erforderlich ist, stellt ein Ansatz mit Webdiensten, der offene Standards wie SOAP, XML und HTTP verwendet, die richtige Wahl dar. Auf der anderen Seite bietet das Remoting die folgenden Funktionen, wenn Sie intranetbasierte Lösungen erstellen, die zwischen Servern verwendet werden:

- Hohe Objekttreue, da für jeden .NET-Typ (einschließlich benutzerdefinierter Typen, die mit den Entwicklungsprogrammen Microsoft C#® und Microsoft Visual Basic® .NET erstellt wurden) das Remoting durchgeführt werden kann. Dies umfasst Klassen, Klassenhierarchien, Schnittstellen, Felder, Eigenschaften, Methoden, Delegates, Datasets, Hashtabellen usw.
- Objekte können nach Wert oder nach Referenz gemarshallt werden.
- Die Verwaltung der Objektlebensdauer basiert auf einem Lease.
- Hohe Leistung, insbesondere mit dem TCP-Kanal und der binären Formatierung.
- Sie ermöglicht es Ihnen, mittlere Ebenen mit Lastenausgleich zu erstellen, wobei Sie den Netzwerklastenausgleich verwenden.

Tabelle 11.2: *Die Hauptunterschiede zwischen dem Remoting und Webdiensten*

Remoting	Webdienste
Statusbehaftete oder statusfreie, auf einem Lease basierende Verwaltung der Objektlebensdauer	Alle Methodenaufrufe sind statusfrei
Kein IIS erforderlich (Obwohl das Hosting in IIS/ASP.NET aus Sicherheitsgründen empfohlen wird)	Auf dem Server muss IIS installiert sein
Es werden alle verwalteten Typen unterstützt	Datentypen werden nur eingeschränkt unterstützt. Weitere Informationen zu diesen von ASP.NET-Webdiensten unterstützten Typen finden Sie in MSDN unter ".NET Framework Developer's Guide" (englischsprachig).
Objekte können als Verweis oder als Wert übergeben werden	Objekte können nicht übergeben werden
Enthält eine erweiterbare Architektur, die nicht auf HTTP- oder TCP-Transporte beschränkt ist	Beschränkt auf XML über HTTP
Benutzerdefinierte Verarbeitungssenken können in die Pipeline zur Nachrichtenverarbeitung einbezogen werden	Keine Möglichkeit zum Ändern von Nachrichten
Die SOAP-Implementierung ist eingeschränkt und es kann nur die RPC-Verschlüsselung verwendet werden	Die SOAP-Implementierung kann die RPC- oder Dokumentenverschlüsselung verwenden und vollständig mit anderen Webdienstplattformen zusammenarbeiten. Weitere Informationen finden Sie in MSDN im Abschnitt "Message Formatting and Encoding" des Artikels "Distributed Application Communication" (englischsprachig).
Enge Bindung	Leichte Bindung

Zusammenfassung

Das .NET Remoting bietet kein eigenes Sicherheitsmodell. Durch das Hosting von Remoteobjekten in ASP.NET und durch Verwenden des HTTP-Kanals für die Kommunikation können Remoteobjekte von den zugrunde liegenden Sicherheitsdiensten profitieren, die von IIS und ASP.NET bereitgestellt werden. Im Vergleich dazu bieten der TCP-Kanal und eine benutzerdefinierte, ausführbare Hostdatei eine höhere Leistung, jedoch keine integrierte Sicherheit.

- Wenn Sie den Client authentifizieren möchten, verwenden Sie den HTTP-Kanal, führen das Hosting in ASP.NET durch und deaktivieren den anonymen Zugriff in IIS.
- Verwenden Sie den TCP-Kanal für bessere Leistungen, wenn die Authentifizierung des Clients für Sie nicht von Bedeutung ist.
- Verwenden Sie IPSec, um den Kommunikationskanal zwischen Client und Server zu sichern, wenn Sie den TCP-Kanal verwenden. Verwenden Sie SSL, um den HTTP-Kanal zu sichern.
- Wenn Sie vertrauenswürdige Aufrufe für Remoteressourcen durchführen müssen, verwalten Sie die Komponente im Windows-Dienst und nicht in einer Konsolenanwendung.
- **IPrincipal**-Objekte werden nicht über .NET Remoting-Grenzen hinweg übermittelt. Sie sollten die Implementierung einer eigenen **IPrincipal**-Klasse erwägen, die serialisiert werden kann. Wenn Sie dies vornehmen, achten Sie darauf, dass es für einen unberechtigten Client relativ einfach ist, ein **IPrincipal**-Objekt nachzuahmen und es an Ihr Remoteobjekt zu senden. Achten Sie auch auf **IlogicalThreadAffinitive**, wenn Sie eine eigene **IPrincipal**-Klasse für das Remoting implementieren.
- Stellen Sie Remoteobjekte niemals im Internet bereit. Verwenden Sie für dieses Szenario Webdienste.

  Das .NET Remoting sollte nur im Intranet verwendet werden. Der Zugriff auf Objekte sollte nur intern über Webanwendungen erfolgen. Auch wenn für das Objekt das Hosting in ASP.NET erfolgt, sollten Sie diese nicht für Internetclients offen legen, da es sich bei den Clients um .NET-Clients handeln müsste.