Erstellen sicherer ASP.NET-Anwendungen

Authentifizierung, Autorisierung und sichere Kommunikation

Kapitel 10 – Webdienstsicherheit

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 hauptsächlich die Sicherheit auf Plattformebene für Webdienste unter Verwendung der zugrunde liegenden Features von IIS und ASP.NET behandelt. Für die Sicherheit auf Nachrichtenebene entwickelt Microsoft das Web Services Development Kit, mit dem Sie sichere Lösungen erstellen können, die die WS-Security-Spezifikation, ein Bestandteil der GXA-Initiative (Global XML Architecture), erfüllen.

Inhalt

In diese Kapitel wird beschrieben, wie die Authentifizierung, Autorisierung und sichere Kommunikationsverfahren entwickelt und angewendet werden, um ASP.NET-Webdienste und Nachrichten von Webdiensten zu sichern. Darin wird die Sicherheit aus Sicht der Webdienste beschrieben sowie gezeigt, wie Aufrufer authentifiziert und autorisiert werden und ein Sicherheitskontext über einen Webdienst übermittelt wird. Außerdem wird aus der Sicht von Clients erläutert, wie Webdienste mit Anmeldeinformationen und Zertifikaten aufgerufen werden, um die serverseitige Authentifizierung zu unterstützen.

Webdienste-Sicherheitsmodell

Die Webdienstsicherheit kann auf drei Ebenen angewendet werden:

- Sicherheit auf der Plattform-/Transportebene (Punkt-zu-Punkt)
- Sicherheit auf Anwendungsebene (benutzerdefiniert)
- Sicherheit auf Nachrichtenebene (Ende-zu-Ende)

Jeder Ansatz besitzt seine Stärken und Schwächen, die nachfolgend sorgfältig ausgeführt werden. Der gewählte Ansatz hängt stark von den Merkmalen der in den Nachrichtenaustausch einbezogenen Architektur und der Plattformen ab.

Hinweis: Beachten Sie, dass sich dieses Kapitel auf die Sicherheit auf Plattform- und Anwendungsebene konzentriert. Mit der Sicherheit auf Nachrichtenebene beschäftigt sich die GXA-Initiative (Global XML Web Services Architecture) und insbesondere die WS-Security-Spezifikation. Zur Zeit der Erstellung dieses Dokuments wurde von Microsoft eine Technologievorschauversion des Web Services Development Kits veröffentlicht. Mit diesem können Sie Sicherheitslösungen für die Nachrichtenebene entwickeln, die der WS-Security-Spezifikation folgen. Weitere Informationen finden Sie unter http://msdn.microsoft.com/webservices/building/wsdk/ (englischsprachig).

Sicherheit auf Plattform-/Transportebene (Punkt-zu-Punkt)

Der Transportkanal zwischen zwei Endpunkten (Webdienstclient und Webdienst) kann verwendet werden, um die Punkt-zu-Punkt-Sicherheit bereitzustellen. Dies wird in Abbildung 10.1 dargestellt.

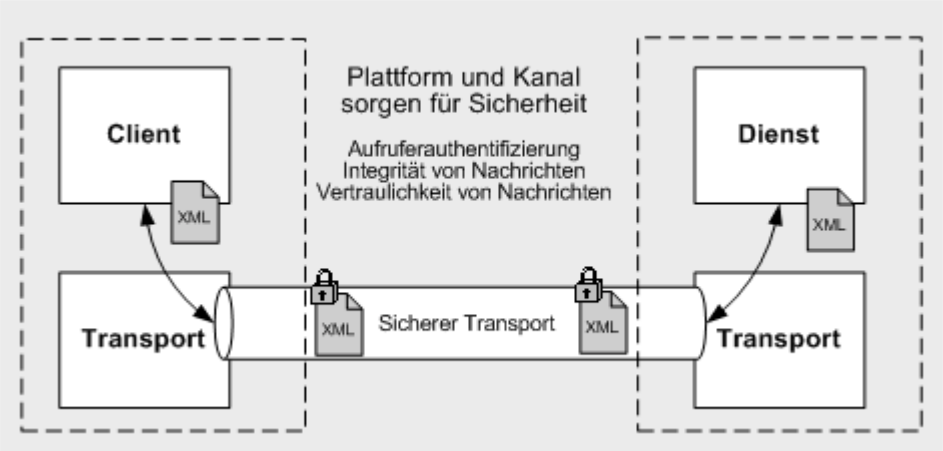

Abbildung 10.1
Sicherheit auf Plattform-/Transportebene

Wenn Sie die Plattformsicherheit, die von einer engen Verbindung mit dem Betriebssystem Microsoft® Windows® ausgeht, z. B. im Unternehmensintranet verwenden, gilt Folgendes:

- Der Webserver (IIS) stellt die Standard-, Digest-, integrierte und Zertifikatsauthentifizierung bereit.
- Der ASP.NET-Webdienst erbt einige der ASP.NET-Authentifizierungs- und Autorisierungsfeatures.
- SSL und/oder IPSec können verwendet werden, um die Integrität und Vertraulichkeit für Nachrichten bereitzustellen.

Verwendungshinweise

Das Modell der Transportebenensicherheit ist einfach, vertraut und für viele (hauptsächlich intranetbasierte) Szenarien geeignet, in denen die Transportmechanismen und die Entpunktkonfiguration umfassend gesteuert werden können.

Die Hauptthemen bei der Transportebenensicherheit sind Folgende:

- Die Sicherheit wird eng an die zugrunde liegende Plattform, den Transportmechanismus sowie den Sicherheitsdienstanbieter (NTLM, Kerberos usw.) gebunden und ist somit von diesen abhängig.
- Die Sicherheit wird auf einer Punkt-zu-Punkt-Basis zugeordnet, ohne Vorkehrungen für mehrere Hops und das Routing über zwischengeschaltete Anwendungsknoten zu bieten.

Sicherheit auf Anwendungsebene

Bei diesem Ansatz ist die Anwendung für die Sicherheit verantwortlich und verwendet benutzerdefinierte Sicherheitsfunktionen. Beispiel:

- Eine Anwendung kann einen benutzerdefinierten SOAP-Header zum Übergeben der Anmeldeinformationen von Benutzern verwenden, damit der Benutzer mit jeder Webdienstanforderung authentifiziert wird. Ein häufiger Ansatz ist die Übergabe eines Tickets (oder eines Benutzernamens bzw. einer Lizenz) im SOAP-Header.
- Die Anwendung verfügt über die Flexibilität, ihr eigenes **IPrincipal**-Objekt zu generieren, das Rollen enthält. Hierbei kann es sich um eine benutzerdefinierte Klasse oder die Klasse **GenericPrincipal** handeln, die vom .NET Framework bereitgestellt wird.
- Die Anwendung kann die erforderlichen Verschlüsselungen selektiv durchführen, obwohl dies die sichere Speicherung von Schlüsseln voraussetzt und die Entwickler die relevanten Crypto APIs kennen müssen.

  Ein alternatives Verfahren stellt die Verwendung von SSL dar, um die Vertraulichkeit und Integrität bereitzustellen und diese zum Durchführen der Authentifizierung mit benutzerdefinierten SOAP-Headern zu verbinden.

Verwendungshinweise

Verwenden Sie diesen Ansatz unter den folgenden Bedingungen:

- Sie möchten den Vorteil eines vorhandenen Datenbankschemas aus Benutzern und Rollen nutzen, das in einer vorhandenen Anwendung verwendet wird.
- Sie möchten eher Teile einer Nachricht verschlüsseln statt des gesamten Datenstroms.

Sicherheit auf Nachrichtenebene (Ende-zu-Ende)

Dies stellt den flexibelsten und leistungsstärksten Ansatz dar, der auch von der GXA-Initiative verwendet wird, insbesondere innerhalb der WS-Security-Spezifikation. Die Sicherheit auf Nachrichtenebene wird in Abbildung 10.2 veranschaulicht.

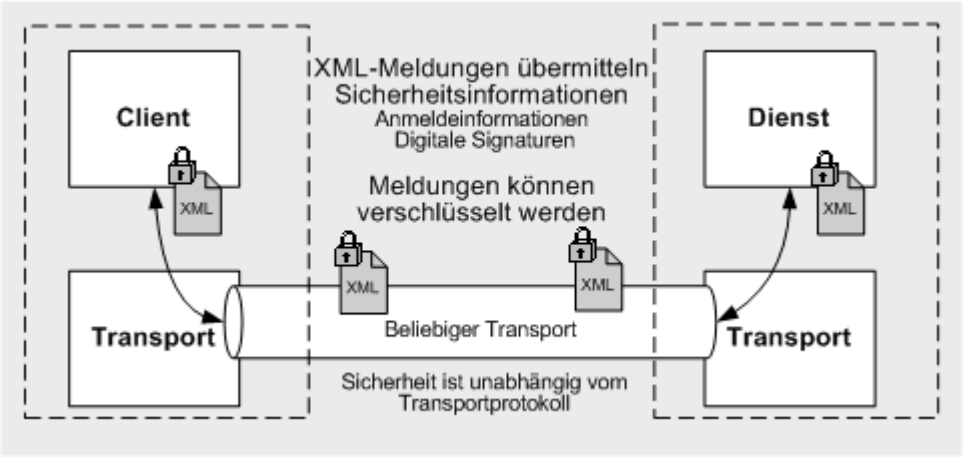

Abbildung 10.2
Sicherheit auf Nachrichtenebene

Die WS-Security-Spezifikationen beschreiben Erweiterungen am SOAP-Messaging, die die Nachrichtenintegrität, Nachrichtenvertraulichkeit und die Authentifizierung von einzelnen Nachrichten bereitstellen.

- Die Authentifizierung wird von Sicherheitstoken bereitgestellt, die in SOAP-Headern übermittelt werden. Bei der WS-Security ist kein bestimmter Tokentyp erforderlich. Die Sicherheitstoken können Kerberos-Tickets, X.509-Zertifikate oder ein benutzerdefiniertes Binärtoken enthalten.
- Die sichere Kommunikation wird durch digitale Signaturen (zum Sicherstellen der Nachrichtenintegrität) und die XML-Verschlüsselung (zum Sicherstellen der Nachrichtenvertraulichkeit) bereitgestellt.

Verwendungshinweise

Die WS-Security kann zum Erstellen eines Frameworks verwendet werden, um sichere Nachrichten in einer heterogenen Webdienstumgebung auszutauschen. Sie ist optimal für heterogene Umgebungen und Szenarien geeignet, in denen Sie die Konfiguration der Endpunkte und zwischengeschalteten Anwendungsknoten nicht direkt steuern.

Sicherheit auf Nachrichtenebene:

- Kann vom zugrunde liegenden Transport unabhängig sein.
- Ermöglicht eine heterogene Sicherheitsarchitektur.
- Bietet durchgehende Sicherheit (Ende-zu-Ende) und ermöglicht das Nachrichtenrouting über zwischengeschaltete Anwendungsknoten.
- Unterstützt mehrere Verschlüsselungsverfahren.
- Unterstützt die Nachweisbarkeit.

Web Services Development Kit

Das Web Services Development Kit stellt zusätzlich zu anderen Diensten (z. B. das Routing und Verweise auf Nachrichtenebene) die zum Verwalten der Sicherheit erforderlichen APIs bereit. Dieses Toolkit entspricht den aktuellsten Webdienststandards (z. B. WS-Security) und ermöglicht daher die Interoperabilität mit anderen Anbietern, die sich nach denselben Spezifikationen richten.

Weitere Informationen

- Die aktuellsten Informationen zum Web Services Development Kit und den WS-Security-Spezifikationen finden Sie in MSDN auf der Seite "XML Developer Center" unter http://msdn.microsoft.com/webservices/ (englischsprachig).
- Weitere Informationen zur WS-Spezifikation finden Sie auf der Indexseite der WS-Security-Spezifikation unter http://msdn.microsoft.com/webservices/default.asp?pull=/library/en-us/dnglobspec/html/wssecurspecindex.asp (englischsprachig).
- Weitere Informationen zu GXA finden Sie in MSDN unter dem Artikel "Understanding GXA" (englischsprachig).
- Diskussionen zu diesem Thema finden Sie in MSDN in der GXA Interoperability Newsgroup (englischsprachig).

Plattform-/Transportsicherheitsarchitektur

- Die Architektur der Plattformsicherheit für ASP.NET-Webdienste ist in Abbildung 10.3 dargestellt.

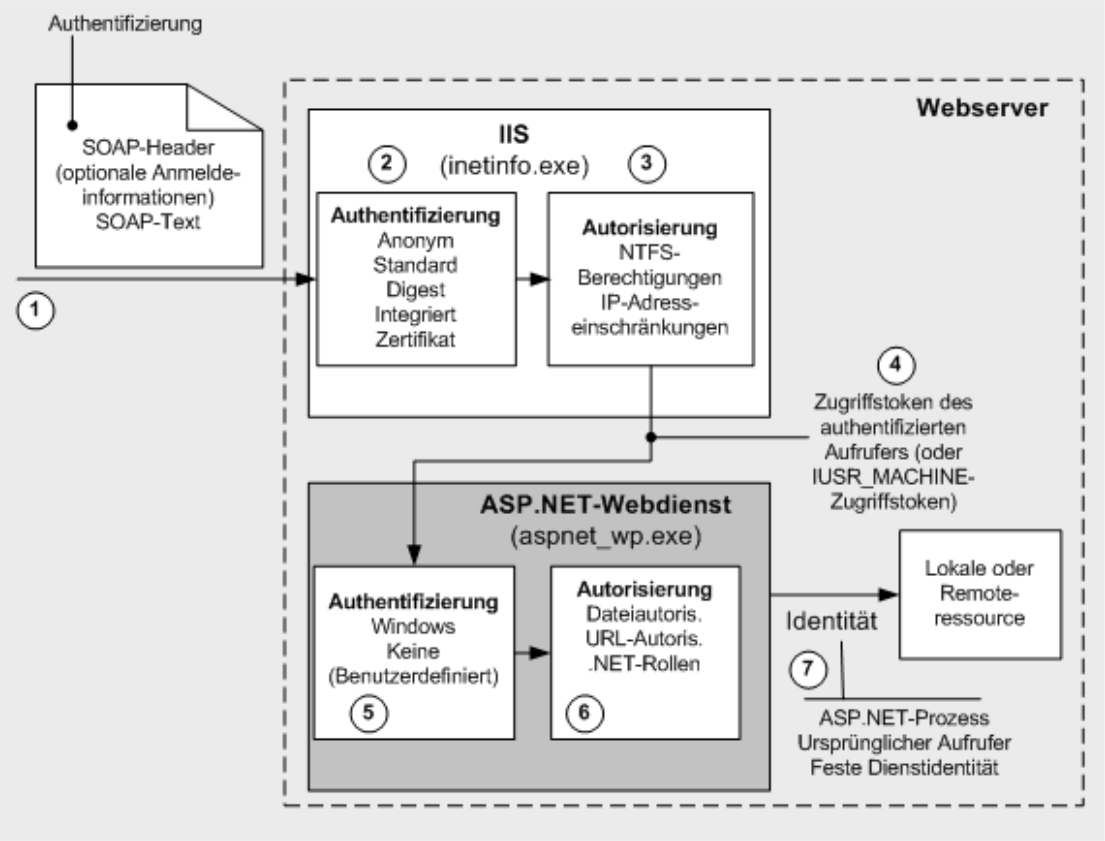

Abbildung 10.3
Sicherheitsarchitektur von Webdiensten

Abbildung 10.3 veranschaulicht die von ASP.NET-Webdiensten bereitgestellten Authentifizierungs- und Autorisierungsmechanismen. Wenn ein Client einen Webdienst aufruft, treten die folgenden Authentifizierungs- und Autorisierungsereignisse ein:

1. Die SOAP-Anfrage wird vom Netzwerk empfangen. Diese kann abhängig vom Typ der verwendeten Authentifizierung möglicherweise Anmeldeinformationen für die Authentifizierung enthalten.
2. IIS authentifiziert den Aufrufer optional über die Standard-, Digest-, integrierte (NTLM oder Kerberos) oder die Zertifikatsauthentifizierung. In heterogenen Umgebungen, in denen die IIS-Authentifizierung (Windows-Authentifizierung) nicht möglich ist, wird IIS für die anonyme Authentifizierung konfiguriert. In diesem Szenario kann der Client unter Verwendung von Attributen der Nachrichtenebene authentifiziert werden, z. B. Tickets, die im SOAP-Header übergeben werden.
3. IIS kann auch so konfiguriert werden, dass nur Anforderungen von Clientcomputern mit bestimmten IP-Adressen akzeptiert werden.
4. IIS übergibt das authentifizierte Windows-Zugriffstoken des Aufrufers an ASP.NET (hierbei kann es sich um das Zugriffstoken des anonymen Internetbenutzers handeln, wenn der Webdienst für die anonyme Authentifizierung konfiguriert ist).

5. Der Aufrufer wird von ASP.NET authentifiziert. Wenn ASP.NET für die Windows-Authentifizierung konfiguriert ist, erfolgt an dieser Stelle keine weitere Authentifizierung. Der Aufrufer wird von IIS authentifiziert.

   Wenn keine Windows-Authentifizierungsmethode verwendet wird, wird für den ASP.NET-Authentifizierungsknoten der Wert **Keine** festgelegt, um die benutzerdefinierte Authentifizierung zu ermöglichen.

   Hinweis: Die Formular- und Passport-Authentifizierung wird momentan nicht für Webdienste unterstützt.

   ---

6. ASP.NET autorisiert den Zugriff auf den angeforderten Webdienst (ASMX-Datei) unter Verwendung der URL- und Dateiautorisierung, die wiederum die der ASMX-Datei zugeordneten NTFS-Berechtigungen verwenden, um zu ermitteln, ob dem authentifizierten Aufrufer der Zugriff gewährt werden soll.

   Hinweis: Die Dateiautorisierung wird nur für die Windows-Authentifizierung unterstützt.

   Sie können für eine feinstufige Autorisierung auch .NET-Rollen verwenden (entweder deklarativ oder programmatisch), um sicherzustellen, dass der Aufrufer für den Zugriff auf die angeforderte Webmethode autorisiert ist.
7. Der Code im Webdienst kann mithilfe einer bestimmten Identität auf lokale und/oder Remoteressourcen zugreifen. Die ASP.NET-Webdienste führen standardmäßig keinen Identitätswechsel durch, weshalb das konfigurierte ASP.NET-Prozesskonto die Identität bereitstellt. Alternative Optionen umfassen die Identität des ursprünglichen Aufrufers oder eine konfigurierte Dienstidentität.

Gatekeeper

Nachfolgend sind die Gatekeeper in einem ASP.NET-Webdienst aufgeführt:

- IIS
  - Wenn die anonyme IIS-Authentifizierung deaktiviert ist, akzeptiert IIS Anforderungen nur von authentifizierten Benutzern.
  - IP-Adresseinschränkungen.

    IIS kann so konfiguriert werden, dass nur Anforderungen von Computern mit bestimmten IP-Adressen akzeptiert werden.
- ASP.NET
  - Das HTTP-Modul für die Dateiautorisierung (nur für die Windows-Authentifizierung).
  - Das HTTP-Modul für die URL-Autorisierung.
- PrincipalPermission-Forderungen und explizite Rollenüberprüfungen.

Weitere Informationen

- Weitere Informationen zu Gatekeepern finden Sie unter "Gatekeeper" in Kapitel 8, "ASP.NET-Sicherheit".
- Weitere Informationen zum Konfigurieren der Sicherheit finden Sie weiter unten in diesem Kapitel unter "Konfigurieren der Sicherheit".

Authentifizierungs- und Autorisierungsstrategien

In diesem Abschnitt wird veranschaulicht, welche Autorisierungsoptionen (konfigurierbar und programmatisch) für eine Reihe häufig verwendeter Authentifizierungsschemas zur Verfügung stehen.

Nachfolgend sind die Authentifizierungsschemas zusammengefasst:

- Windows-Authentifizierung mit Identitätswechsel
- Windows-Authentifizierung ohne Identitätswechsel
- Windows-Authentifizierung mit fester Identität

Windows-Authentifizierung mit Identitätswechsel

Die folgenden Konfigurationselemente zeigen Ihnen, wie die Windows-Authentifizierung (IIS) und der Identitätswechsel deklarativ in der Datei **Web.config** oder **Machine.config** aktiviert werden.

Hinweis: Sie sollten die Authentifizierung auf Webdienstbasis für die einzelnen Webdienste in der Datei **Web.config** konfigurieren.

```
<authentication mode="Windows" />
<identity impersonate="true" />
```

Bei dieser Konfiguration nimmt der Webdienstcode die Identität des für IIS authentifizierten Aufrufers an. Sie müssen den anonymen Zugriff in IIS deaktivieren, um für den ursprünglichen Aufrufer den Identitätswechsel durchzuführen. Beim anonymen Zugriff führt der Webdienstcode einen Identitätswechsel für das anonyme Internetbenutzerkonto durch (bei dem es sich standardmäßig um IUSR_MACHINE handelt).

Konfigurierbare Sicherheit

Wenn Sie die Windows-Authentifizierung zusammen mit dem Identitätswechsel verwenden, stehen Ihnen die folgenden Autorisierungsoptionen zur Verfügung:

- **Windows Access Control Lists (ACLs, Zugriffssteuerungslisten)**
  - **Webdienstdatei (ASMX-Dateien)** – Bei der Dateiautorisierung werden für angeforderte ASP.NET-Ressourcen (einschließlich der ASMX-Webdienstdatei) mithilfe des Sicherheitskontextes des ursprünglichen Aufrufers Zugriffsüberprüfungen durchgeführt. Der ursprüngliche Aufrufer muss mindestens über den Lesezugriff auf die ASMX-Datei verfügen.
  - **Ressourcen, auf die der Webdienst zugreift** – Windows-Zugriffssteuerungslisten, auf die der Webdienst zugreift (Dateien, Ordner, Registrierungsschlüssel, Active Directory®-Verzeichnisobjekte usw.), müssen einen Access Control Entry (ACE, Zugriffssteuerungseintrag) einbeziehen, der dem ursprünglichen Aufrufer den Lesezugriff gewährt (da der Identitätswechsel für den Aufrufer vom Webdienstthread durchgeführt wird, mit dem der Ressourcenzugriff erfolgt).

- **URL-Autorisierung** – Diese ist in der Datei **Machine.config** und/oder **Web.config** konfiguriert. Bei der Windows-Authentifizierung nehmen Benutzernamen die Form **Domänenname\Benutzername** an, während Rollen den Windows-Gruppen 1:1 zugeordnet werden.

```
<authorization>
  <deny user="DomainName\UserName" />
  <allow roles="DomainName\WindowsGroup" />
</authorization>
```

Programmatische Sicherheit

Die programmatische Sicherheit bezieht sich auf Sicherheitsüberprüfungen, die sich im Webdienstcode befinden. Die folgenden programmatischen Sicherheitsoptionen stehen zur Verfügung, wenn Sie die Windows-Authentifizierung und den Identitätswechsel verwenden:

- **PrincipalPermission-Forderungen**
  - Imperativ (inline im Code einer Methode enthalten)

```
PrincipalPermission permCheck = new PrincipalPermission(
                                    null, @"DomainName\WindowsGroup");
permCheck.Demand();
```

  - Deklarativ (diese Attribute können vor Webmethoden oder Webklassen eingesetzt werden)

```
// Demand that the caller is a member of a specific role (for Windows
// authentication this is the same as a Windows group)
[PrincipalPermission(SecurityAction.Demand,
                     Role=@"DomainName\WindowsGroup")]
// Demand that the caller is a specific user
[PrincipalPermission(SecurityAction.Demand,
                     Name=@"DomainName\UserName")]
```

- **Explizite Rollenüberprüfungen** – Sie können Rollenüberprüfungen mithilfe der Schnittstelle **IPrincipal** durchführen.

```
IPrincipal.IsInRole(@"DomainName\WindowsGroup");
```

Verwendungshinweise

Verwenden Sie die Windows-Authentifizierung und den Identitätswechsel, wenn Folgendes zutrifft:

- Die Clients des Webdienstes können mithilfe von Windows-Konten identifiziert werden, die vom Server authentifiziert werden können.
- Sie müssen den Sicherheitskontext des ursprünglichen Aufrufers über den Webdienst auf die nächste Ebene übermitteln. Beispielsweise an eine Reihe von Serviced Components, die Enterprise Services (COM+)-Rollen verwenden oder an eine Datenebene, die eine feinstufige Autorisierung erfordert (auf Benutzerbasis).
- Sie müssen den Sicherheitskontext des ursprünglichen Aufrufers durch die nachgeschalteten Ebenen leiten, um die Überwachung auf Betriebssystemebene zu unterstützen.

Wichtig: Die Verwendung des Identitätswechsels kann die Skalierbarkeit verringern, da er sich auf das Datenbankverbindungspooling auswirkt. Als alternativen Ansatz können Sie die Verwendung des Modells mit dem vertrauenswürdigen Subsystem erwägen, bei dem der Webdienst die Aufrufer autorisiert und dann eine feste Identität für den Datenbankzugriff verwendet. Sie können die Identität des Aufrufers auf der Anwendungsebene übermitteln, z. B. durch Verwenden von Parametern gespeicherter Prozeduren.

Weitere Informationen

- Weitere Informationen zur Windows-Authentifizierung und zum Identitätswechsel finden Sie in Kapitel 8, "ASP.NET-Sicherheit".
- Weitere Informationen zur URL-Autorisierung finden Sie unter "URL-Autorisierungsknoten" in Kapitel 8, "ASP.NET-Sicherheit".

Windows-Authentifizierung ohne Identitätswechsel

Die folgenden Konfigurationselemente zeigen Ihnen, wie die Windows-Authentifizierung (IIS) ohne Identitätswechsel deklarativ in der Datei **Web.config** aktiviert wird.

```
<authentication mode="Windows" />
<!-- The following setting is equivalent to having no identity element -->
<identity impersonate="false" />
```

Konfigurierbare Sicherheit

Wenn Sie die Windows-Authentifizierung ohne Identitätswechsel verwenden, stehen Ihnen die folgenden Autorisierungsoptionen zur Verfügung:

- **Windows-Zugriffssteuerungslisten**
  - **Webdienstdatei (ASMX-Dateien)** – Bei der Dateiautorisierung werden für angeforderte ASP.NET-Ressourcen (einschließlich der ASMX-Webdienstdatei) mithilfe des ursprünglichen Aufrufers Zugriffsüberprüfungen durchgeführt. Es ist kein Identitätswechsel erforderlich.
  - **Ressourcen, auf die die Anwendung zugreift** – Windows-Zugriffssteuerungslisten auf Ressourcen, auf die Ihre Anwendung zugreift (Dateien, Ordner, Registrierungsschlüssel, Active Directory-Objekte) müssen einen ACE einbeziehen, der der ASP.NET-Prozessidentität (die vom Webdienstthread für den Zugriff auf Ressourcen verwendete Standardidentität) den Lesezugriff gewährt.
- **URL-Autorisierung**
  Diese ist in der Datei **Machine.config** und **Web.config** konfiguriert. Bei der Windows-Authentifizierung nehmen Benutzernamen die Form **Domänenname\Benutzername** an, während Rollen den Windows-Gruppen 1:1 zugeordnet werden.

```
<authorization>
  <deny user="DomainName\UserName" />
  <allow roles="DomainName\WindowsGroup" />
</authorization>
```

Programmatische Sicherheit

Die programmatische Sicherheit bezieht sich auf Sicherheitsüberprüfungen, die sich im Webdienstcode befinden. Die folgenden programmatischen Sicherheitsoptionen stehen zur Verfügung, wenn Sie die Windows-Authentifizierung ohne Identitätswechsel verwenden:

- **PrincipalPermission-Forderungen**
  - Imperativ

```
PrincipalPermission permCheck = new PrincipalPermission(
                                    null, @"DomainName\WindowsGroup");
permCheck.Demand();
```

  - Deklarativ

```
// Demand that the caller is a member of a specific role (for Windows
// authentication this is the same as a Windows group)
[PrincipalPermission(SecurityAction.Demand,
                     Role=@"DomainName\WindowsGroup")]
// Demand that the caller is a specific user
[PrincipalPermission(SecurityAction.Demand,
                Name=@"DomainName\UserName")]
```

- **Explizite Rollenüberprüfungen** – Sie können die Rollenüberprüfung mithilfe der Schnittstelle **IPrincipal** durchführen.

```
IPrincipal.IsInRole(@"DomainName\WindowsGroup");
```

Verwendungshinweise

Verwenden Sie die Windows-Authentifizierung ohne Identitätswechsel, wenn Folgendes zutrifft:

- Die Clients des Webdienstes können mithilfe von Windows-Konten identifiziert werden, die vom Server authentifiziert werden können.
- Sie möchten das Modell des vertrauenswürdigen Subsystems verwenden und Clients innerhalb des Webdienstes autorisieren, um dann eine feste Identität für den Zugriff auf nachgeschaltete Ressourcen zu verwenden (z. B. Datenbanken), damit das Verbindungspooling unterstützt wird.

Weitere Informationen

- Weitere Informationen zur Windows-Authentifizierung und zum Identitätswechsel finden Sie in Kapitel 8, "ASP.NET-Sicherheit".
- Weitere Informationen zur URL-Autorisierung finden Sie unter "URL-Autorisierungsknoten" in Kapitel 8, "ASP.NET-Sicherheit".

Windows-Authentifizierung mit fester Identität

Das Element <**identity**> in der Datei **Web.config** unterstützt optionale Benutzernamen- und Kennwortattribute, die es Ihnen ermöglichen, eine bestimmte feste Identität für den Identitätswechsel für Ihren Webdienst zu konfigurieren. Dies wird im folgenden Fragment der Konfigurationsdatei veranschaulicht.

```
<identity impersonate="true" userName="DomainName\UserName"
                             password="ClearTextPassword" />
```

Verwendungshinweise

Dieser Ansatz wird für sichere Umgebungen aus zwei Gründen nicht empfohlen:

- Benutzernamen und Kennwörter sollten in Konfigurationsdateien nicht unverschlüsselt gespeichert werden.
- Unter Windows 2000 zwingt Sie dieser Ansatz, dem ASP.NET-Prozesskonto das Recht **Als Teil des Betriebssystems handeln** zu gewähren. Dadurch wird die Sicherheit der Webanwendung verringert und das Risiko erhöht, wenn ein Angreifer den Webdienstprozess (**Aspnet_wp.exe**) gefährdet.

Weitere Informationen

- Weitere Informationen zur Windows-Authentifizierung und zum Identitätswechsel finden Sie in Kapitel 8, "ASP.NET-Sicherheit".
- Weitere Informationen zur URL-Autorisierung finden Sie unter "URL-Autorisierungsknoten" in Kapitel 8, "ASP.NET-Sicherheit".

Konfigurieren der Sicherheit

In diesem Abschnitt werden die praktischen Schritte veranschaulicht, die zum Konfigurieren der Sicherheit für einen ASP.NET-Webdienst erforderlich sind. Diese sind in Abbildung 10.4 zusammengefasst.

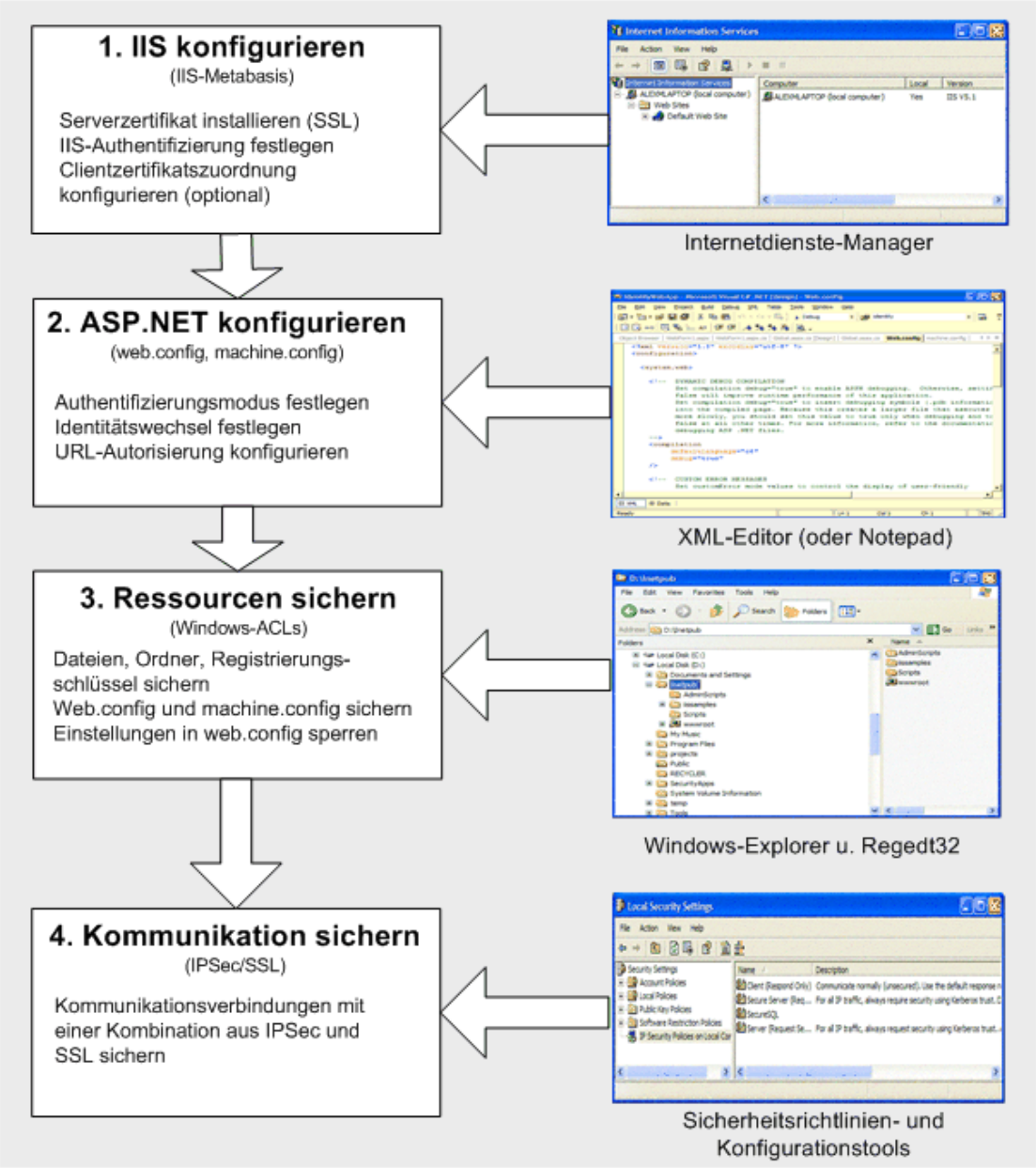

Abbildung 10.4
Konfigurieren der ASP.NET-Webdienstsicherheit

Konfigurieren der IIS-Einstellungen

Ausführliche Informationen zum Konfigurieren der IIS-Sicherheitseinstellungen finden Sie unter "Konfigurieren der Sicherheit" in Kapitel 8, "ASP.NET-Sicherheit", da sich diese Informationen auch auf ASP.NET-Webdienste beziehen.

Konfigurieren der ASP.NET-Einstellungen

Die Konfigurationseinstellungen der Anwendungsebene werden in **Web.config**-Dateien gespeichert, die sich im virtuellen Stammverzeichnis Ihres Webdienstes befinden. Konfigurieren Sie die folgenden Einstellungen:

1. **Konfigurieren der Authentifizierung** – Dies sollte auf Webdienstbasis in der Datei **Web.config** festgelegt werden (nicht in **Machine.config**), die sich im virtuellen Stammverzeichnis des Webdienstes befindet.

   ```
   <authentication mode="Windows|None" />
   ```

   Hinweis: Webdienste unterstützen momentan keine Passport- oder Formularauthentifizierung. Bei der benutzerdefinierten Authentifizierung oder der Authentifizierung auf Nachrichtenebene legen Sie für den Modus den Wert **Keine** fest.

2. **Konfigurieren des Identitätswechsels und der Autorisierung** – Ausführliche Informationen finden Sie unter "Konfigurieren der Sicherheit" in Kapitel 8, "ASP.NET-Sicherheit".

Weitere Informationen

Weitere Informationen zur URL-Autorisierung finden Sie unter "URL-Autorisierungsknoten" in Kapitel 8, "ASP.NET-Sicherheit".

Sichere Ressourcen

Sie sollten zum Sichern der Webressourcen dieselben Verfahren verwenden, die in Kapitel 8, "ASP.NET-Sicherheit", vorgestellt werden. Zusätzlich sollten Sie für Webdienste in Betracht ziehen, das Protokoll HTTP-GET und HTTP-POST auf Produktionsservern aus der Datei **Machine.config** zu entfernen.

Deaktivieren von HTTP-GET, HTTP-POST

Clients können standardmäßig mit ASP.NET-Webdiensten unter Verwendung von drei Protokollen kommunizieren: HTTP-GET, HTTP-POST und SOAP über HTTP. Sie sollten die Unterstützung sowohl für das Protokoll HTTP-GET als auch für HTTP-POST für Produktionscomputer auf Computerebene deaktivieren, auf denen diese nicht erforderlich sind. Dadurch wird eine potenzielle Sicherheitsverletzung vermieden, die es einer unberechtigten Webseite ermöglichen kann, auf einen internen Webdienst zuzugreifen, der hinter einem Firewall ausgeführt wird.

Hinweis: Das Deaktivieren dieser Protokolle bedeutet, dass ein neuer Client nicht in der Lage ist, einen XML-Webdienst über die Schaltfläche **Aufrufen** auf der Testseite des Webdienstes zu testen. Stattdessen müssen Sie ein Testclientprogramm durch Hinzufügen eines Verweises auf einen Webdienst (unter Verwendung des Entwicklungssystems Microsoft Visual Studio® .NET) erstellen. Auf Entwicklungscomputern können diese Protokolle aktiviert bleiben, um es Entwicklern zu ermöglichen, die Testseite zu verwenden.

▶ **So deaktivieren Sie die Protokolle HTTP-GET und HTTP-POST für einen gesamten Computer:**

1. Bearbeiten Sie die Datei **Machine.config**.
2. Versehen Sie die Zeilen im Element <**webServices**> mit Kommentarzeichen, die die Unterstützung für HTTP-GET und HTTP-POST hinzufügen. Anschließend sollte die Datei **Machine.config** wie folgt aussehen:

```
<webServices>
    <protocols>
      <add name="HttpSoap"/>
         <!-- <add name="HttpPost"/> -->
         <!-- <add name="HttpGet"/>  -->
      <add name="Documentation"/>
    </protocols>
</webServices>
```

3. Speichern Sie die Datei **Machine.config**.

Hinweis: In besonderen Fällen, bei denen Sie Webdienstclients besitzen, die mit einem Webdienst über HTTP-GET oder HTTP-POST kommunizieren, können Sie die Unterstützung für diese Protokolle zur Datei **Web.config** der Anwendung hinzufügen, indem Sie das Element <**webServices**> erstellen und die Unterstützung für diese Protokolle mit den Elementen <**protocol**> und <**add**> hinzufügen, wie bereits zuvor gezeigt.

Weitere Informationen

Weitere Informationen zum Sichern von Ressourcen finden Sie unter "Sichern von Ressourcen" in Kapitel 8, "ASP.NET-Sicherheit".

Sichere Kommunikation

Verwenden Sie eine Kombination aus SSL und IPSec, um die Kommunikationsverbindungen zu sichern.

Weitere Informationen

- Weitere Informationen zum Aufrufen von Webdiensten unter Verwendung von SSL finden Sie unter "Vorgehensweise: Aufrufen eines Webdienstes unter Verwendung von SSL" im Abschnitt "Referenz" dieses Handbuchs.
- Weitere Informationen zum Verwenden von IPSec zwischen zwei Computern finden Sie unter "Vorgehensweise: Verwenden von IPSec zum Sichern der Kommunikation zwischen zwei Servern" im Abschnitt "Referenz" dieses Handbuchs.

Übergeben von Anmeldeinformationen an Webdienste zur Authentifizierung

Wenn Sie einen Webdienst aufrufen, erfolgt dies über einen Webdienstproxy, einem lokalen Objekt, das denselben Satz an Methoden offen legt wie der Zielwebdienst.

Sie können einen Webdienstproxy mit dem Befehlszeilenprogramm **Wsdl.exe** generieren. Alternativ können Sie bei Verwendung von Visual Studio .NET den Proxy durch Hinzufügen eines Webverweises zum Projekt hinzufügen.

Hinweis: Wenn der Webdienst, für den Sie einen Proxy generieren möchten, so konfiguriert ist, dass er Clientzertifikate erfordert, müssen Sie diese Anforderung temporär deaktivieren, während Sie den Verweis hinzufügen. Andernfalls tritt ein Fehler auf. Nachdem Sie den Verweis hinzugefügt haben, müssen Sie den Dienst erneut umkonfigurieren, damit er wieder Zertifikate erfordert.

Einen alternativen Ansatz stellt das Bereitstellen einer WSDL (Web Services Description Language)-Offlinedatei für benutzereigene Anwendungen dar. Sie müssen daran denken, diese zu aktualisieren, wenn sich die Webdienstschnittstelle ändert.

Festlegen von Clientanmeldeinformationen für die Windows-Authentifizierung

Wenn Sie die Windows-Authentifizierung verwenden, müssen Sie die Anmeldeinformationen festlegen, die für die Authentifizierung unter Verwendung der Eigenschaft **Credentials** des Webdienstproxys verwendet wird. Wenn Sie diese Eigenschaft nicht explizit festlegen, wird der Webdienst 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"

Clientanmeldeinformationen werden nicht implizit übermittelt. Der Webdienstnutzer muss die Anmeldeinformationen und Authentifizierungsdetails für den Proxy festlegen. Sie können die Eigenschaft **Credentials** des Webdienstproxys, wie unten gezeigt, auf den Wert **CredentialCache**.**DefaultCredentials** festlegen, um den Sicherheitskontext vom Windows-Sicherheitskontext des Clients an einen Webdienst zu übermitteln (entweder von einem Identitätswechsel-Threadtoken oder einem Prozesstoken).

```
proxy.Credentials = System.Net.CredentialCache.DefaultCredentials;
```

Bedenken Sie die folgenden Punkte, bevor Sie diesen Ansatz verwenden:

- Dadurch werden die Clientanmeldeinformationen nur übermittelt, wenn Sie die NTLM-, Kerberos- oder Verhandlungsauthentifizierung verwenden.
- Wenn eine clientseitige Anwendung (z. B. eine Windows-Formularanwendung) den Webdienst aufruft, werden die Anmeldeinformationen von der interaktiven Anmeldesitzung des Benutzers abgerufen.
- Serverseitige Anwendungen, z. B. ASP.NET-Webanwendungen, verwenden die Prozessidentität, wenn nicht der Identitätswechsel konfiguriert wurde. In diesem Fall wird die Identität des Aufrufers nach dem Identitätswechsel verwendet.

Verwenden von bestimmten Anmeldeinformationen

Verwenden Sie den folgenden Code, um einen bestimmten Anmeldeinformationensatz für die Authentifizierung zu verwenden, wenn Sie einen Webdienst aufrufen:

```
CredentialCache cache = new CredentialCache();
cache.Add( new Uri(proxy.Url), // Web service URL
           "Negotiate",        // Kerberos or NTLM
           new NetworkCredential("username", "password", "domainname") );
proxy.Credentials = cache;
```

Im o. a. Beispiel führt der angeforderte Verhandlungsauthentifizierungstyp entweder zur Kerberos- oder NTLM-Authentifizierung.

Immer einen bestimmten Authentifizierungstyp anfordern

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

```
proxy.Credentials = new
                      NetworkCredential("username", "password", "domainname");
```

Dies sollte im Produktionscode vermieden werden, da Sie keine Kontrolle über den vom Webdienst 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, stattdessen jedoch 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, um bestimmte Authentifizierungsanmeldeinformationen bereitzustellen, die dazu führen, dass mit der Webanforderung der HTTP-Header **WWW-authenticate** übergeben wird. Dadurch muss der Webserver weder den Zugriff für die Anforderung ablehnen noch die Authentifizierung für die nachfolgende Wiederholungsanforderung durchführen.

Hinweis: Eine Vorauthentifizierung wird nur angewendet, nachdem der Webdienst das erste Mal erfolgreich authentifiziert wurde. Die Vorauthentifizierung hat keine Auswirkungen auf die erste Webanforderung.

```
private void ConfigureProxy( WebClientProtocol proxy,
                             string domain, string username,
                             string password )
{
  // To improve performance, force pre-authentication
  proxy.PreAuthenticate = true;
  // Set the credentials
  CredentialCache cache = new CredentialCache();
  cache.Add( new Uri(proxy.Url),
             "Negotiate",
             new NetworkCredential(username, password, domain) );
  proxy.Credentials = cache;
  proxy.ConnectionGroupName = username;
}
```

Verwenden der Eigenschaft "ConnectionGroupName"

Beachten Sie, dass der o. a. Code die Eigenschaft **ConnectionGroupName** des Webdienstproxys festlegt. Dies ist nur erforderlich, wenn der zum Verbinden mit dem Webdienst verwendete Sicherheitskontext zwischen den einzelnen Anforderungen unterschiedlich ist, wie nachfolgend beschrieben.

Wenn Sie eine ASP.NET-Webanwendung besitzen, die eine Verbindung zu einem Webdienst herstellt und den Sicherheitskontext des ursprünglichen Aufrufers übermittelt (durch Aufrufen von **DefaultCredentials** oder durch Festlegen von expliziten Anmeldeinformationen, wie zuvor gezeigt), sollten Sie die Eigenschaft **ConnectionGroupName** des Webdienstproxys innerhalb der Webanwendung festlegen. Dadurch wird verhindert, dass ein neuer, nicht authentifizierter Client eine bestehende, authentifizierte TCP-Verbindung zum Webdienst wiederverwenden kann, die den Authentifizierungsanmeldeinformationen eines vorherigen Clients zugeordnet sind. 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, wie im vorherigen Codefragment gezeigt.

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

Aufrufen der Webdienste von Clients, die nicht Windows ausführen

Es gibt eine Reihe von Authentifizierungsansätzen, die für browserübergreifende Szenarien geeignet sind. Dabei handelt es sich um die Folgenden:

- **Zertifikatsauthentifizierung** – Verwenden von plattformübergreifenden X.509-Zertifikaten.
- **Standardauthentifizierung** – Ein Beispiel zum Verwenden der Standardauthentifizierung für einen benutzerdefinierten Datenspeicher (ohne Active Directory) finden Sie unter http://www.rassoc.com/gregr/weblog/stories/2002/06/26/webServicesSecurityHttpBasicAuthenticationWithoutActiveDirectory.html (englischsprachig).
- **GXA-Nachrichtenebenenansätze** – Verwenden Sie das Web Services Development Toolkit, um GXA-Lösungen (WS-Security) zu implementieren.
- **Benutzerdefinierte Ansätze** – Beispielsweise das Übermitteln von Anmeldeinformationen in SOAP-Headern.

Proxyserverauthentifizierung

Die Proxyserverauthentifizierung wird nicht vom Dialogfeld **Webverweis hinzufügen** von Visual Studio .NET unterstützt (obwohl sie in der nächsten Version von Visual Studio .NET unterstützt wird). Stattdessen erhalten Sie möglicherweise einen HTTP-Status 407, **Proxyauthentifizierung erforderlich**, wenn Sie versuchen, einen Webverweis hinzuzufügen.

Hinweis: Dieser Fehler wird möglicherweise nicht angezeigt, wenn Sie die ASMX-Datei in einem Browser anzeigen, da der Browser automatisch Anmeldeinformationen sendet.

Sie können das Befehlszeilenprogramm **Wsdl.exe** verwenden (statt des Dialogfelds **Webverweis hinzufügen**), um dieses Problem zu umgehen, wie nachfolgend gezeigt:

```
wsdl.exe /proxy:http://<YourProxy> /pu:<YourName> /pp:<YourPassword>
/pd:<YourDomain> http://www.YouWebServer.com/YourWebService/YourService.asmx
```

Wenn Sie die Daten für die Proxyserverauthentifizierung programmgesteuert festlegen müssen, verwenden Sie den folgenden Code:

```
YourWebServiceProxy.Proxy.Credentials = CredentialsCache.DefaultCredentials;
```

Übermitteln des ursprünglichen Aufrufers

In diesem Abschnitt wird beschrieben, wie Sie den Sicherheitskontext des ursprünglichen Aufrufers über eine ASP.NET-Webanwendung an einen Webdienst übermitteln können, der sich auf einem Remoteanwendungsserver befindet. Sie müssen dies möglicherweise durchführen, um innerhalb des Webdienstes oder innerhalb der nachgeschalteten Subsysteme (z. B. Datenbanken, bei denen ursprüngliche Aufrufer für einzelne Datenbankobjekte autorisiert werden sollen) die Autorisierung auf Benutzerbasis zu unterstützen.

In Abbildung 10.5 wird der Sicherheitskontext des ursprünglichen Aufrufers (Alice) 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.

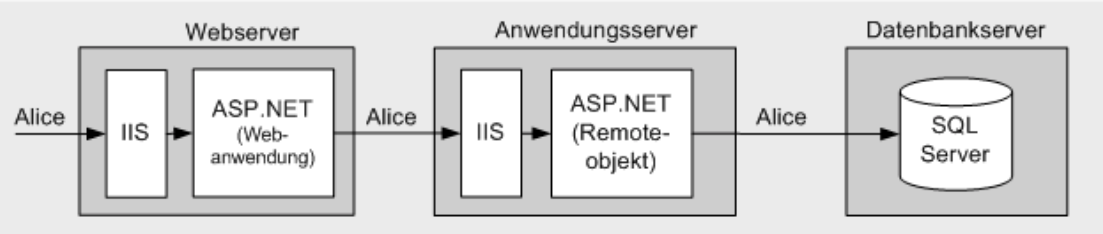

Abbildung 10.5
Übermitteln des Sicherheitskontextes des ursprünglichen Aufrufers

Damit die Anmeldeinformationen an einen Webdienst übermittelt werden, muss der Webdienstclient (in diesem Szenario die ASP.NET-Webanwendung) den Webdienstproxy konfigurieren und die Eigenschaft **Credentials** des Proxys explizit festlegen, wie weiter oben in diesem Kapitel unter "Übergeben von Anmeldeinformationen für die Authentifizierung an Webdienste" beschrieben wurde.

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 Webdienstproxy 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 höhere Version ausführen, um die Kerberos-Delegierung verwenden zu können. Zusätzlich müssen Clientkonten, die delegiert werden sollen, in 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 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 Webdienstproxys	
Schritt	**Weitere Informationen**
Festlegen der Eigenschaft **Credentials** des Webdienstproxys auf den Wert **DefaultCredentials**	Ein Codebeispiel hierzu finden Sie weiter oben in diesem Kapitel unter "Verwenden von DefaultCredentials".

Konfigurieren des Remoteanwendungsservers

Konfigurieren von IIS	
Schritt	**Weitere Informationen**
Deaktivieren des anonymen Zugriffs für das virtuelle Stammverzeichnis des Webdienstes	
Aktivieren der integrierten Windows-Authentifizierung für das virtuelle Stammverzeichnis der Webanwendung	

Konfigurieren von ASP.NET (Webdiensthost)	
Schritt	Weitere Informationen
Konfigurieren von ASP.NET, sodass die Windows-Authentifizierung verwendet wird	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis des Webdienstes. Legen Sie das <**authentication**>-Element fest auf: `<authentication mode="Windows" />`
Konfigurieren Sie ASP.NET für den Identitätswechsel	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis des Webdienstes. Legen Sie das <**identity**>-Element fest auf: `<identity impersonate="true" />` **Hinweis**: Dieser Schritt ist nur erforderlich, wenn der Sicherheitskontext des ursprünglichen Aufrufers über den Webdienst 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 Webdienst auf Benutzerbasis ermöglichen möchten, müssen Sie hier keinen Identitätswechsel durchführen.

Weitere Informationen

Weitere Informationen zum Konfigurieren der 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 den Webdienst verwenden.

Bei diesem Ansatz stehen die unverschlüsselten Anmeldeinformationen des Clients für die Webanwendung zur Verfügung. Diese können über den Webdienstproxy an einen Webdienst übergeben werden. Dazu müssen Sie Code in der Webanwendung schreiben, um die Anmeldeinformationen des Clients abzurufen und den Proxy 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 Webdienstproxys 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 Web service proxy
// To improve performance, force preauthentication
proxy.PreAuthenticate = true;
// Set the credentials
CredentialCache cache = new CredentialCache();
cache.Add( new Uri(proxy.Url),
           "Basic",
           new NetworkCredential(uid, pwd, domain) );
proxy.Credentials = cache;
```

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 Web service proxy
// To improve performance, force preauthentication
proxy.PreAuthenticate = true;
// Set the credentials
CredentialCache cache = new CredentialCache();
cache.Add( new Uri(proxy.Url),
           "Basic",
           new NetworkCredential(uid, pwd, domain) );
proxy.Credentials = cache;
```

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, das kein <**identity**>-Element vorhanden ist. Der Identitätswechsel ist nicht erforderlich, da die Anmeldeinformationen des Clients explizit über den Proxy an den Webdienst übergeben werden.

Konfigurieren des Webdienstproxys	
Schritt	Weitere Informationen
Schreiben von Code, um die Anmeldeinformationen für den Webdienstproxy zu erfassen und explizit festzulegen	Informationen hierzu finden Sie in den zuvor gezeigten Codefragmenten, die in den Abschnitten "Standardauthentifizierung" und "Formularauthentifizierung" enthalten sind.

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 (des Webdienstes) ermöglicht es dem Webdienst, 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 den Webdienst 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 dem Webdienst nicht zur Verfügung.

Konfigurieren von ASP.NET (Webdienst)	
Schritt	Weitere Informationen
Konfigurieren von ASP.NET, sodass die Windows-Authentifizierung verwendet wird	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis des Webdienstes. Legen Sie das <**authentication**>-Element fest auf: `<authentication mode="Windows" />`
Konfigurieren des ASP.NET-Webdienstes für Identitätswechsel	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis des Webdienstes. Legen Sie das <**identity**>-Element fest auf: `<identity impersonate="true" />` **Hinweis**: Dieser Schritt ist nur erforderlich, wenn der Sicherheitskontext des ursprünglichen Aufrufers über den Webdienst 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 Webdienst 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 Webdienst und der Webanwendung eine Vertrauensgrenze. Der Webdienst vertraut der Webanwendung, die Aufrufer ordnungsgemäß zu authentifizieren und zu autorisieren, bevor sie Anforderungen an den Webdienst durchlässt. Beim Webdienst erfolgt keine Authentifizierung des ursprünglichen Aufrufers. Der Webdienst authentifiziert die feste vertraute Identität, die von der Webanwendung zum Kommunizieren mit dem Webdienst verwendet wird. In den meisten Fällen ist dies die Prozessidentität des ASP.NET-Workerprozesses.

Das Modell mit dem vertrauten Subsystem ist in Abbildung 10.6 dargestellt.

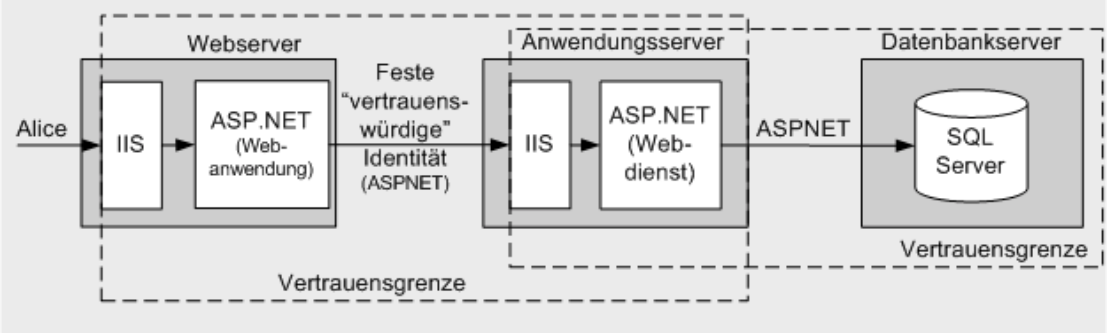

Abbildung 10.6
Das Modell mit vertrauenswürdigen Subsystemen

Übermitteln der Aufruferidentität

Wenn Sie das vertrauenswürdige 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 = "Alice"
```

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 Webdienste) 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 Webdienstproxys	
Schritt	Weitere Informationen
Konfigurieren des Webdienstproxys für die Verwendung von Standardanmeldeinformationen für alle Aufrufe des Webdienstes	Verwenden Sie die folgende Codezeile: `proxy.Credentials = DefaultCredentials;`

Konfigurieren des Anwendungsservers

Konfigurieren von IIS	
Schritt	Weitere Informationen
Deaktivieren des anonymen Zugriffs für das virtuelle Stammverzeichnis des Webdienstes Aktivieren der integrierten Windows-Authentifizierung	

Konfigurieren von ASP.NET	
Schritt	Weitere Informationen
Konfigurieren von ASP.NET, sodass die Windows-Authentifizierung verwendet wird	Bearbeiten Sie **Web.config** im virtuellen Verzeichnis des Webdienstes. 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" />`

Zugreifen auf Systemressourcen

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

Zugreifen auf Netzwerkressourcen

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

- Die Prozessidentität (vom Konto bestimmt, das zum Ausführen des ASP.NET-Workerprozesses verwendet wird)
- Eine Dienstidentität (z. B. durch Aufrufen von **LogonUser** erstellt)
- Die Identität des ursprünglichen Aufrufers (mit dem Webdienst für den Identitätswechsel konfiguriert)

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

Zugreifen auf COM-Objekte

Die Direktive **AspCompat** (wird von Webanwendungen verwendet, wenn Sie COM-Apartmentthreadobjekte aufrufen) steht nicht für Webdienste zur Verfügung. Das bedeutet, wenn Sie Apartmentmodellobjekte über Webdienste aufrufen, erfolgt ein Threadwechsel. Dies führt zu einem leichten Leistungsrückgang, und wenn der Webdienst einen Identitätswechsel durchführt, geht das Identitätswechseltoken beim Aufrufen der COM-Komponente verloren. Dies führt normalerweise zu einer Ausnahmebedingung, bei der der Zugriff verweigert wird.

Weitere Informationen

- Weitere Informationen zu Ausnahmebedingungen, bei denen der Zugriff verweigert wird, finden Sie in der Microsoft Knowledge Base im Artikel Q325791,"PRB: Access Denied Error Message Occurs When Impersonating in ASP.NET and Calling STA COM Components" (US).
- Weitere Informationen zum Zugreifen auf COM-Objekte und Verwenden des **AspCompat**-Attributs finden Sie unter "Zugreifen auf COM-Objekte" in Kapitel 8, "ASP.NET-Sicherheit".
- Weitere Informationen zum Aufrufen von COM-Apartmentthreadobjekten finden Sie in der Microsoft Knowledge Base im Artikel Q303375, "INFO: XML Web Services and Apartment Objects" (US).

Verwenden von Clientzertifikaten mit Webdiensten

In diesem Abschnitt werden Verfahren zum Verwenden der X.509-Clientzertifikate für die Webdienstauthentifizierung beschrieben.

Sie können innerhalb eines Webdienstes die Clientzertifikatsauthentifizierung verwenden, um Folgendes zu authentifizieren:

- Andere Webdienste.
- Anwendungen, die direkt mit dem Webdienst kommunizieren (z. B. serverbasierte oder clientseitige Desktopanwendungen).

Authentifizieren von Webbrowserclients mit Zertifikaten

Ein Webdienst kann keine Clientzertifikate verwenden, um Aufrufer zu authentifizieren, wenn diese mit einer zwischengelagerten Webanwendung interagieren, da es nicht möglich ist, das Zertifikat des ursprünglichen Aufrufers über die Webanwendung an einen Webdienst weiterzuleiten. Während die Webanwendung ihre Clients mit Zertifikaten authentifizieren kann, können dieselben Zertifikate dann nicht vom Webdienst zur Authentifizierung verwendet werden.

Der Grund, dass dieses Server-zu-Server-Szenario fehlschlägt, ist, dass die Webanwendung keinen Zugriff auf das Clientzertifikat (insbesondere auf den privaten Schlüssel) in ihrem Zertifikatspeicher hat. Dieses Problem wird in Abbildung 10.7 dargestellt.

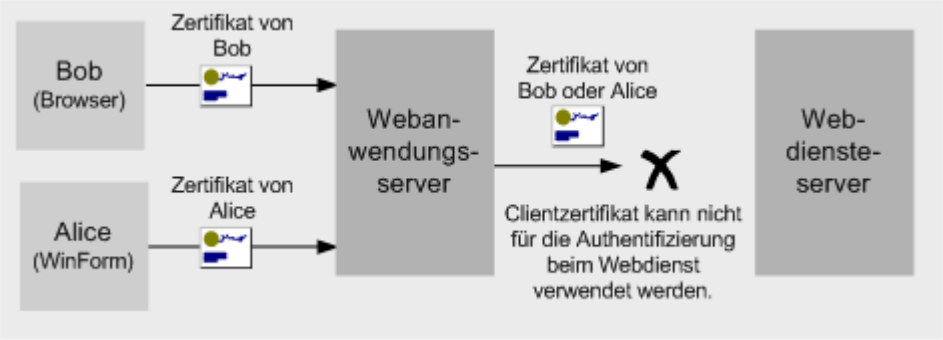

Abbildung 10.7
Webdienst-Clientzertifikatsauthentifizierung

Verwenden des Modells mit vertrauenswürdigen Subsystemen

Sie müssen ein Modell mit vertrauenswürdigem Subsystem verwenden, um diese Einschränkung zu behandeln und die Zertifikatsauthentifizierung zu unterstützen. Mit diesem Ansatz authentifiziert der Webdienst die Webanwendung mithilfe des Zertifikats der Webanwendung (und nicht mit dem Zertifikat des ursprünglichen Aufrufers). Der Webdienst muss der Webanwendung vertrauen, um deren Benutzer zu authentifizieren sowie die erforderliche Autorisierung durchzuführen, um sicherzustellen, dass nur autorisierte Aufrufer auf die vom Webdienst offen gelegten Daten und Funktionen zugreifen können.

Diese Vorgehensweise ist in Abbildung 10.8 dargestellt.

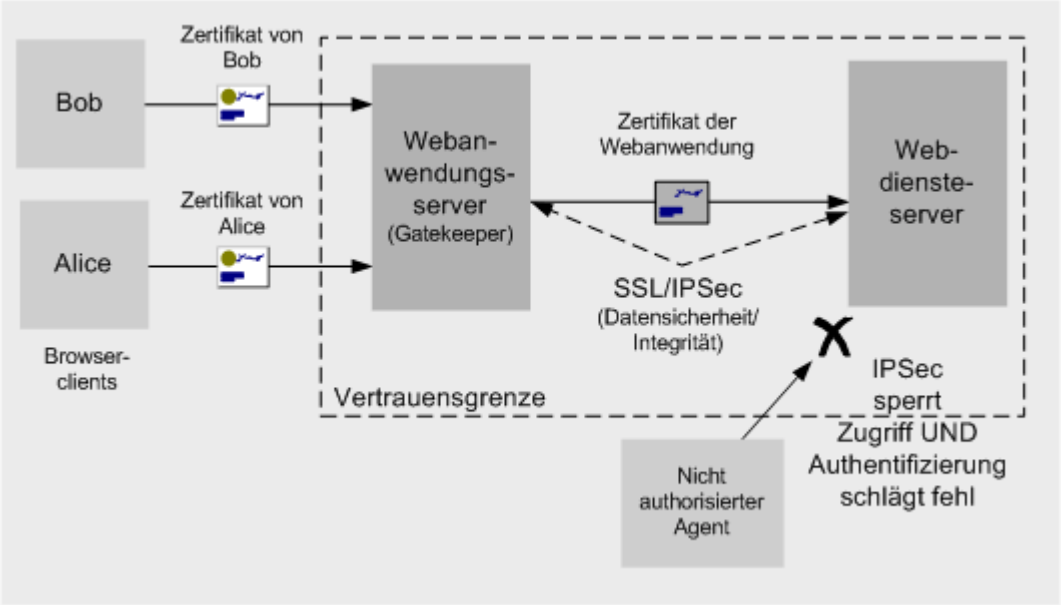

Abbildung 10.8
Der Webdienst authentifiziert die vertrauenswürdige Webanwendung

Wenn die Autorisierungslogik im Webdienst mehrere Rollen erfordert, kann die Webanwendung verschiedene Zertifikate senden, die auf der Rollenmitgliedschaft des Aufrufers basieren. Beispielsweise kann ein Zertifikat für Mitglieder einer Administratorengruppe (denen es möglich ist, Daten innerhalb einer Back-End-Datenbank zu aktualisieren) und ein anderes Zertifikat für alle anderen Benutzer (die nur für Lesevorgänge autorisiert sind) verwendet werden.

Hinweis: In derartigen Szenarien kann ein lokaler Zertifikatserver verwendet werden (wobei der Zugriff nur den beiden Servern gewährt wird), um alle Webanwendungszertifikate zu verwalten.

In diesem Szenario gilt:

- Die Webanwendung authentifiziert ihre Benutzer unter Verwendung von Clientzertifikaten.
- Die Webanwendung fungiert als Gatekeeper, autorisiert ihre Benutzer und steuert den Zugriff auf den Webdienst.
- Die Webanwendung ruft den Webdienst auf und übergibt ein anderes Zertifikat, das die Anwendung darstellt (oder möglicherweise einen Zertifikatbereich, der auf der Rollenmitgliedschaft des Aufrufers basiert).
- Der Webdienst authentifiziert die Webanwendung und vertraut der Anwendung bei der Durchführung der erforderlichen Clientautorisierung.

- Zwischen dem Webanwendungsserver und dem Webdienstserver wird IPSec verwendet, um eine zusätzliche Zugriffssteuerung bereitzustellen. Durch IPSec werden unautorisierte Zugriffsversuche durch andere Computer verhindert. Die Zertifikatsauthentifizierung auf dem Webdienstserver verhindert ebenfalls den unautorisierten Zugriff.

Lösungsimplementierung

Verwenden Sie einen separaten Prozess zum Aufrufen des Webdienstes und zum Übergeben des Zertifikats, um die Zertifikatsauthentifizierung für den Webdienst in diesem Szenario zu verwenden. Sie können die Zertifikate nicht direkt über die ASP.NET-Webanwendung ändern, da diese nicht über ein geladenes Benutzerprofil und einen zugeordneten Zertifikatspeicher verfügt. Der separate Prozess muss so konfiguriert werden, dass er mithilfe eines Kontos ausgeführt wird, dem ein Benutzerprofil (und Zertifikatspeicher) zugeordnet ist. Ihnen bieten sich zwei hauptsächliche Optionen:

- Verwenden einer Enterprise Services-Serveranwendung
- Verwenden eines Windows-Dienstes

Abbildung 10.9 veranschaulicht dieses Szenario mit einer Enterprise Services-Serveranwendung.

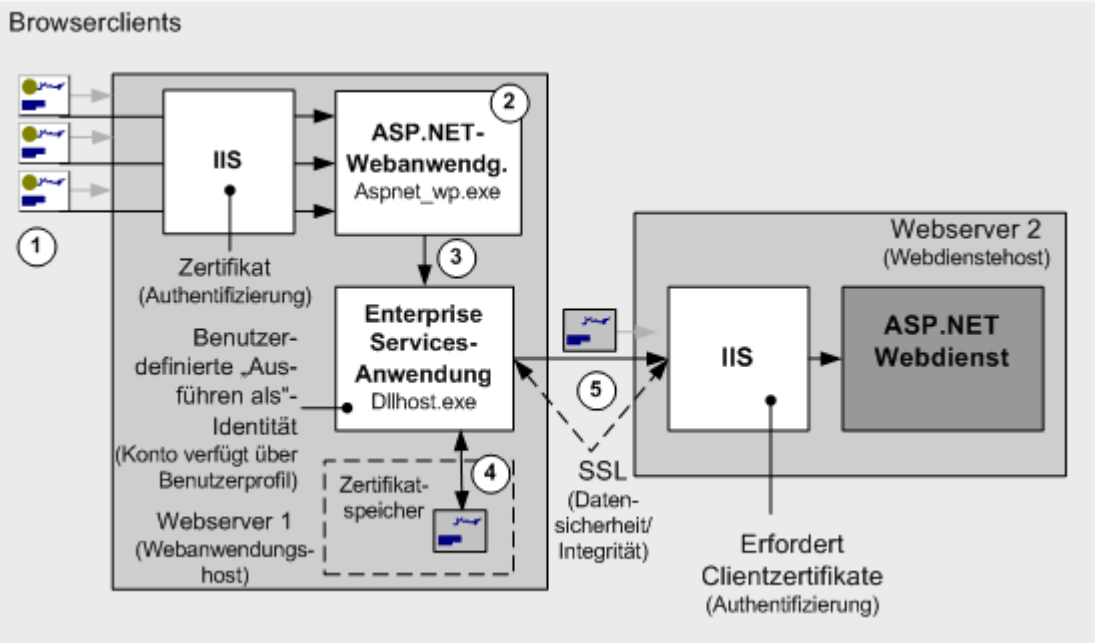

Abbildung 10.9
Clientzertifikatsauthentifizierung mit Webdiensten

Nachfolgend wird die in Abbildung 10.9 dargestellte Ereignisabfolge zusammengefasst:

1. Der ursprüngliche Aufrufer wird von der Webanwendung unter Verwendung von Clientzertifikaten authentifiziert.
2. Die Webanwendung ist der Gatekeeper und verantwortlich für das Autorisieren des Zugriffs auf bestimmte Funktionsbereiche (einschließlich der Bereiche, die die Interaktion mit dem Webdienst einbeziehen).
3. Die Webanwendung ruft eine Serviced Component auf, die in einer prozessexternen Enterprise Services-Serveranwendung ausgeführt wird.
4. Dem zum Ausführen der Enterprise Services-Anwendung verwendeten Konto ist ein Benutzerprofil zugeordnet. Die Komponente greift auf ein Clientzertifikat aus ihrem Zertifikatspeicher zu, das vom Webdienst zum Authentifizieren der Webanwendung verwendet wird.

5. Die Serviced Component ruft den Webdienst auf und übergibt das Clientzertifikat an jede Methodenanforderung. Der Webdienst authentifiziert die Webanwendung unter Verwendung dieses Zertifikats und vertraut der Webanwendung beim ordnungsgemäßen Autorisieren von ursprünglichen Aufrufern.

Gründe zum Verwenden eines zusätzlichen Prozesses

Ein zusätzlicher Prozess ist aufgrund der Tatsache erforderlich (anstatt den Webprozess **Aspnet_wp.exe** zu verwenden, um den Kontakt zum Webdienst herzustellen), dass ein Benutzerprofil erforderlich ist (das einen Zertifikatspeicher enthält).

Eine Webanwendung, die unter Verwendung des Kontos ASPNET ausgeführt wird, besitzt keinen Zugriff auf Zertifikate auf dem Webserver. Der Grund hierfür ist, dass Zertifikatspeicher innerhalb von Benutzerprofilen verwaltet werden, die interaktiven Benutzerkonten zugeordnet sind. Benutzerprofile werden nur für interaktive Konten erstellt, wenn Sie sich physisch mit dem Konto anmelden. Das Konto ASPNET ist nicht als interaktives Benutzerkonto gedacht und wird mit der Berechtigung zum Verweigern der interaktiven Anmeldung konfiguriert, um zusätzliche Sicherheit zu erreichen.

Wichtig: Konfigurieren Sie das Konto ASPNET nicht erneut, um dieses Recht zu entfernen und das Konto in ein interaktives Anmeldekonto zu ändern. Verwenden Sie einen separaten Prozess mit einem konfigurierten Dienstkonto, um auf Zertifikate zuzugreifen, wie bereits in diesem Kapitel beschrieben.

Weitere Informationen

- Weitere Informationen zum Implementieren dieses Ansatzes finden Sie unter "Vorgehensweise: Aufrufen eines Webdienstes mit Clientzertifikaten von ASP.NET" im Abschnitt "Referenz" dieses Handbuchs.
- Weitere Informationen zum Konfigurieren von IPSec finden Sie unter "Vorgehensweise: Verwenden von IPSec zum Sichern der Kommunikation zwischen zwei Servern" im Abschnitt "Referenz" dieses Handbuchs.

Sichere Kommunikation

Die sichere Kommunikation befasst sich mit der garantierten Integrität und Vertraulichkeit von Webdienstnachrichten, während diese über das Netzwerk zwischen Anwendungen übermittelt werden. Diesem Problem widmen sich zwei Ansätze: Transportebenenoptionen und Nachrichtenebenenoptionen.

Transportebenenoptionen

Die Transportebenenoptionen umfassen:

- SSL
- IPSec

Diese Optionen sind möglicherweise angemessen, wenn die folgenden Bedingungen vorliegen:

- Sie senden eine Nachricht direkt von Ihrer Anwendung an einen Webdienst, und die Nachricht wird nicht über zwischengeschaltete Systeme geleitet.
- Sie können die Konfiguration beider in die Nachrichtenübertragung einbezogenen Endpunkte steuern.

Nachrichtenebenenoptionen

Nachrichtenebenenansätze können verwendet werden, um die Vertraulichkeit und Integrität von Nachrichten zu gewährleisten, während diese über eine beliebige Anzahl zwischengeschalteter Systeme übertragen werden. Nachrichten können signiert werden, um die Integrität bereitzustellen. Zum Erreichen der Vertraulichkeit können Sie zwischen dem Verschlüsseln der gesamten Nachricht und eines Teils der Nachricht wählen.

Verwenden Sie einen Nachrichtenebenenansatz, wenn die folgenden Bedingungen zutreffen:

- Sie senden eine Nachricht an einen Webdienst, und die Nachricht wird sehr wahrscheinlich an andere Webdienste weitergeleitet oder über zwischengeschaltete Systeme geleitet.
- Sie steuern die Konfiguration beider Endpunkte nicht, da Sie z. B. Nachrichten von einem Unternehmen an ein anderes senden.

Weitere Informationen

- Das Web Service Development Toolkit stellt gemäß der WS-Security-Spezifikation Verschlüsselungsfunktionen für Nachrichten bereit.
- Weitere Informationen zu SSL und IPSec finden Sie in Kapitel 4, "Sichere Kommunikation".

Zusammenfassung

Dieses Kapitel konzentriert sich auf die Webdienstsicherheit auf Plattform-/Transportebene (Punkt-zu-Punkt), die von zugrunde liegenden Diensten wie ASP.NET, IIS und dem Betriebssystem bereitgestellt werden. Während die Sicherheit auf Plattformebene sichere Lösungen für eng verbundene Intranetszenarien bereitstellt, ist sie nicht für heterogene Szenarien geeignet. Dazu ist die Sicherheit auf Nachrichtenebene erforderlich, die von der WS-Security-Spezifikation der GXA bereitgestellt wird. Verwenden Sie das Web Services Development Kit, um Sicherheitslösungen für Webdienste auf Nachrichtenebene zu erstellen.

Für eng verbundene Umgebungen mit Windows-Domänen:

- Wenn Sie die Identität des ursprünglichen Aufrufers von einer ASP.NET-Webanwendung an einen Remotewebdienst übermitteln möchten, sollte die ASP.NET-Webanwendung die Kerberos-Authentifizierung (mit Konten, die für die Delegierung konfiguriert sind), Standardauthentifizierung oder Formularauthentifizierung verwenden.
    - Bei der Kerberos-Authentifizierung aktivieren Sie den Identitätswechsel mit der Webanwendung und konfigurieren die Eigenschaft **Credentials** des Webdienstproxys unter Verwendung von **DefaultCredentials**.
    - Bei der Standard- oder Formularauthentifizierung erfassen Sie die Anmeldeinformationen des Aufrufers und legen die Eigenschaft **Credentials** des Webdienstproxys fest, indem Sie ein neues **CredentialCache**-Objekt hinzufügen.

Für Szenarien, bei denen die Übermittlung von einem Webdienst an einen anderen Webdienst erfolgt:

- Verwenden Sie die Standard- oder Kerberos-Authentifizierung, und setzen Sie die Anmeldeinformationen im Clientproxy fest.
- Verwenden Sie eine prozessexterne Enterprise Services-Anwendung oder einen Windows-Dienst, um X.509-Zertifikate von Webanwendungen zu ändern.
- Verwenden Sie, soweit möglich, Autorisierungsüberprüfungen auf Systemebene (z. B. die Datei- und URL-Autorisierung).
- Für eine detaillierte Autorisierung (z. B. auf Webmethodenebene) verwenden Sie .NET-Rollen (deklarativ und imperativ).
- Autorisieren Sie Nicht-Windows-Benutzer unter Verwendung von .NET-Rollen (basierend auf einem **GenericPrincipal**-Objekt, das Rollen enthält).
- Deaktivieren Sie auf Produktservern die Protokolle HTTP-GET und HTTP-POST.
- Verwenden Sie die Transportebenensicherheit, wenn Sie sich keine Gedanken darüber machen, ob die Nachrichten sicher über zwischengeschaltete Systeme übertragen werden.
- Verwenden Sie die Transportebenensicherheit, wenn die SSL-Leistung akzeptabel ist.
- Verwenden Sie die WS-Security und das Web Services Development Kit, um Lösungen auf Nachrichtenebene zu entwickeln.