Erstellen sicherer ASP.NET-Anwendungen

Authentifizierung, Autorisierung und sichere Kommunikation

Kapitel 8 – ASP.NET-Sicherheit

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 werden Anleitungen und Empfehlungen behandelt, die beim Erstellen sicherer ASP.NET-Webanwendungen hilfreich sind. Viele der in diesem Kapitel bereitgestellten Anleitungen und Empfehlungen gelten auch für die Entwicklung von ASP.NET-Webdiensten und .NET Remoting-Objekten, die von ASP.NET verwaltet werden.

Inhalt

ASP.NET-Sicherheitsarchitektur

ASP.NET arbeitet mit IIS, dem .NET Framework sowie den vom Betriebssystem bereitgestellten, zugrunde liegenden Sicherheitsdiensten zusammen, um eine Reihe von Authentifizierungs- und Autorisierungsmechanismen zu bieten. Diese sind in Abbildung 8.1 zusammengefasst.

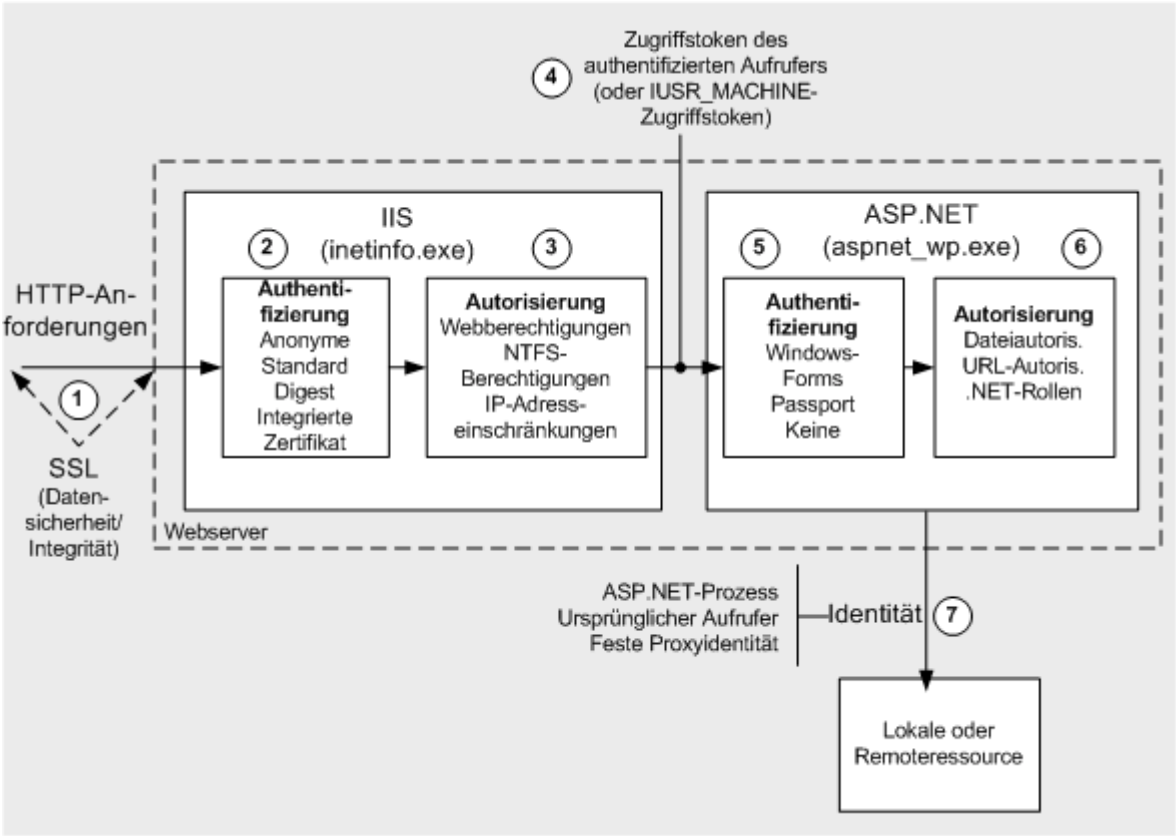

Abbildung 8.1
ASP.NET-Sicherheitsdienste

In Abbildung 8.1 werden die Authentifizierungs- und Autorisierungsmechanismen veranschaulicht, die von IIS und ASP.NET bereitgestellt werden. Wenn ein Client eine Webanfrage startet, treten die folgenden Authentifizierungs- und Autorisierungsereignisse ein:

1. Die HTTP(S)-Webanfrage wird vom Netzwerk empfangen. Sie können SSL verwenden, um die Serveridentität (mithilfe von Serverzertifikaten) sowie wahlweise die Clientidentität zu gewährleisten.

Hinweis: SSL bietet auch einen sicheren Kanal, um vertrauliche Daten zu schützen, die zwischen Client und Server (und umgekehrt) übertragen werden.

---

2. IIS authentifiziert den Aufrufer über die Standard-, Digest-, integrierte (NTLM oder Kerberos) oder die Zertifikatsauthentifizierung. Wenn die gesamte oder ein Teil der Site keinen authentifizierten Zugriff erfordert, kann IIS für die anonyme Authentifizierung konfiguriert werden. IIS erstellt für jeden authentifizierten Benutzer ein Windows-Zugriffstoken. Wenn die anonyme Authentifizierung gewählt wurde, erstellt IIS für das anonyme Internetbenutzerkonto (standardmäßig IUSR_MACHINE) ein Zugriffstoken.
3. IIS autorisiert den Aufrufer für den Zugriff auf die angeforderten Ressourcen. Um den Zugriff zu autorisieren, werden durch Access Control Lists (ACL, Zugriffssteuerungsliste) definierte NTFS-Berechtigungen verwendet, die der angeforderten Ressource zugeordnet sind. 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 die anonyme Authentifizierung verwendet wird).
5. Der Aufrufer wird von ASP.NET authentifiziert.

   Wenn ASP.NET für die Windows-Authentifizierung konfiguriert ist, erfolgt an dieser Stelle keine weitere Authentifizierung. ASP.NET akzeptiert dann jedes von IIS erhaltene Token.

   Wenn ASP.NET für die Formularauthentifizierung konfiguriert ist, werden die vom Aufrufer bereitgestellten Anmeldeinformationen anhand eines Datenspeichers authentifiziert, bei dem es sich normalerweise um eine Microsoft® SQL Server™-Datenbank oder einen Active Directory®-Verzeichnisdienst handelt. Wenn ASP.NET für die Passport-Authentifizierung konfiguriert ist, wird der Benutzer an eine Passport-Website weitergeleitet, wo er von einem Passport-Authentifizierungsdienst authentifiziert wird.
6. ASP.NET autorisiert den Zugriff auf die angeforderte Ressource oder Operation.

   UrlAuthorizationModule (ein vom System bereitgestelltes HTTP-Modul) verwendet in der Datei **Web.config** konfigurierte Autorisierungsregeln (insbesondere das Element <**authorization**>), um sicherzustellen, dass der Aufrufer auf die angeforderte Datei oder den Ordner zugreifen kann.

   Bei der Windows-Authentifizierung prüft **FileAuthorizationModule** (ein weiteres HTTP-Modul), ob der Aufrufer über die erforderlichen Berechtigungen verfügt, um auf die angeforderte Ressource zugreifen zu können. Das Zugriffstoken des Aufrufers wird mit der Zugriffssteuerungsliste verglichen, die diese Ressource schützt.

   Sie können auch .NET-Rollen verwenden (entweder deklarativ oder programmatisch), um sicherzustellen, dass der Aufrufer für den Zugriff auf die angeforderte Ressource oder die Durchführung der angeforderten Operation autorisiert ist.
7. Der Code in der Anwendung greift mithilfe einer bestimmten Identität auf lokale und/oder Remoteressourcen zu. ASP.NET führt standardmäßig keinen Identitätswechsel durch, weshalb das konfigurierte ASP.NET-Prozesskonto die Identität bereitstellt. Alternative Optionen umfassen die Identität des ursprünglichen Aufrufers (falls der Identitätswechsel aktiviert ist) oder eine konfigurierte Dienstidentität.

Gatekeeper

Die Autorisierungspunkte (oder Gatekeeper) in einer ASP.NET-Anwendung werden von IIS und ASP.NET bereitgestellt:

IIS

Bei deaktivierter anonymer Authentifizierung gestattet IIS nur Anforderungen von Benutzern, die entweder in der eigenen oder in einer vertrauenswürdigen Domäne authentifiziert werden können.

Bei statischen Dateitypen (z. B. JPG-, GIF- und HTM-Dateien, d. h. Dateien, die nicht einer ISAPI-Erweiterung zugeordnet sind) verwendet IIS die NTFS-Berechtigungen zur Zugriffssteuerung, die der angeforderten Datei zugeordnet sind.

ASP.NET

Die ASP.NET-Gatekeeper umfassen **UrlAuthorizationModule**, **FileAuthorizationModule** sowie PrincipalPermission-Forderungen und Rollenüberprüfungen.

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

Für Dateitypen, die von IIS der ASP.NET ISAPI-Erweiterung (**Aspnet_isapi.dll**) zugeordnet sind, werden anhand des authentifizierten Windows-Zugriffstokens des Benutzers (z. B. IUSR_MACHINE) automatische Zugriffsüberprüfungen mit der Zugriffssteuerungsliste durchgeführt, die der angeforderten ASP.NET-Datei zugeordnet ist.

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

Die Klasse **FileAuthorizationModule** führt Zugriffsüberprüfungen nur für die angeforderte Datei und nicht für Dateien durch, auf die der Code in der angeforderten Seite zugreift, obwohl für diese eine Zugriffsüberprüfung durch IIS erfolgt.

Wenn Sie z. B. die Datei **Default.aspx** anfordern und diese ein eingebettetes Benutzersteuerelement (**Usercontrol.ascx**) enthält, das wiederum ein Bildtag (Verweis auf **Image.gif**) einbezieht, führt **FileAuthorizationModule** eine Zugriffsüberprüfung für **Default.aspx** und **Usercontrol.ascx** durch, da diese Dateitypen von IIS der ASP.NET ISAPI-Erweiterung zugeordnet sind.

FileAuthorizationModule führt keine Überprüfung für **Image.gif** durch, da es sich hierbei um eine statische Datei handelt, die von IIS intern behandelt wird. Da Zugriffsüberprüfungen für statische Dateien von IIS durchgeführt werden, muss dem authentifizierten Benutzer weiterhin mit einer ordnungsgemäß konfigurierten Zugriffssteuerungsliste die Leseberechtigung für die Datei erteilt werden.

Dieses Szenario ist in Abbildung 8.2 dargestellt.

Hinweis für Systemadministratoren: Der authentifizierte Benutzer muss für alle im Szenario einbezogenen Dateien die NTFS-Leseberechtigungen besitzen. Die einzige Variable stellt der Gatekeeper dar, der zum Durchsetzen der Zugriffssteuerung gewählt wird. Das ASP.NET-Prozesskonto erfordert lediglich den Lesezugriff auf die in ASP.NET registrierten Dateitypen.

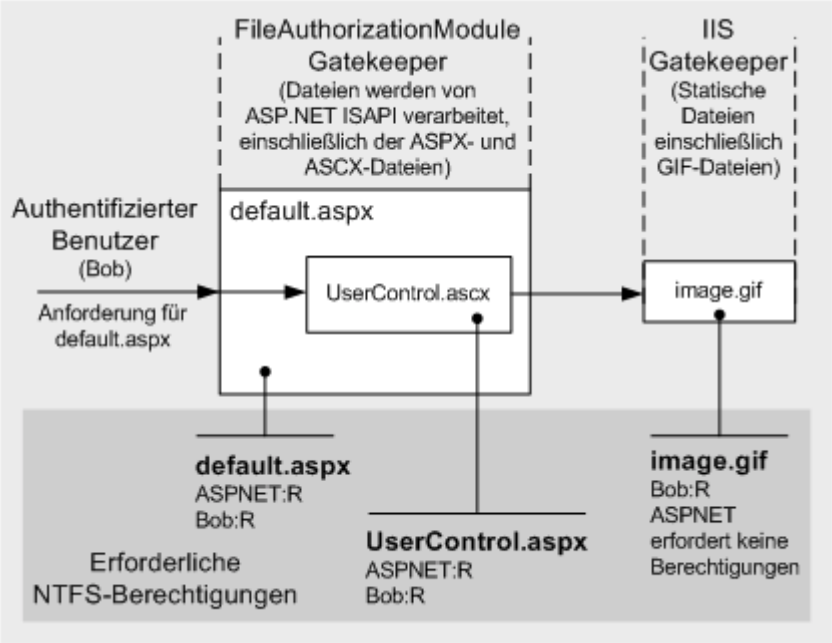

Abbildung 8.2
Zusammenarbeit der IIS- und ASP.NET-Gatekeeper

In diesem Szenario können Sie den Zugriff am Dateigate verhindern. Wenn Sie die der Datei **Default.aspx** zugeordnete Zugriffssteuerungsliste konfigurieren und den Zugriff für einen bestimmten Benutzer verweigern, werden weder das Benutzersteuerelement, noch eingebettete Bilder über den Code in **Default.aspx** an den Client gesendet. Wenn der Benutzer die Bilder direkt anfordert, führt IIS die Zugriffsüberprüfungen durch.

PrincipalPermission-Forderungen und explizite Rollenüberprüfungen

Zusätzlich zu den konfigurierbaren IIS- und ASP.NET-Gatekeepern können Sie auch PrincipalPermission-Forderungen (deklarativ oder programmatisch) als zusätzlichen feinstufigen Zugriffssteuerungsmechanismus verwenden. Principalberechtigungsüberprüfungen (von der Klasse **PrincipalPermissionAttribute** durchgeführt) 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: Die zum Anfordern der Rollenmitgliedschaft verwendeten PrincipalPermission-Forderungen unterscheiden sich vom Aufrufen von **IPrincipal.IsInRole** zum Testen der Rollenmitgliedschaft. Ersteres führt zu einer Ausnahmebedingung, wenn der Aufrufer kein Mitglied der angegebenen Rolle ist, während im letzteren Fall einfach ein boolescher Wert zurückgegeben wird, 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. Bei der Formular- und Passport-Authentifizierung wird das Objekt **GenericPrincipal** mit der entsprechenden Identität und ohne Rollen erstellt und **HttpContext.User** zugeordnet.

Weitere Informationen

- Weitere Informationen zum Konfigurieren der Sicherheit finden Sie weiter unten in diesem Kapitel unter "Konfigurieren der Sicherheit".
- Weitere Informationen zum Programmieren der Sicherheit (und über **IPrincipal**-Objekte) finden Sie weiter unten in diesem Kapitel unter "Programmieren der Sicherheit".

Authentifizierungs- und Autorisierungsstrategien

ASP.NET bietet eine Reihe von deklarativen und programmatischen Autorisierungsmechanismen, die zusammen mit einer Vielzahl von Authentifizierungsschemas verwendet werden können. Dies ermöglicht es Ihnen, eine ausführliche Autorisierungsstrategie sowie eine Strategie zu entwickeln, die zum Bereitstellen verschiedener Feinstufigkeitsgrade konfiguriert werden kann, z. B. auf Benutzer- oder Benutzergruppenbasis (rollenbasiert).

In diesem Abschnitt wird veranschaulicht, welche Autorisierungsoptionen (sowohl konfigurierbar als auch programmatisch) für eine Reihe häufig verwendeter Authentifizierungsoptionen zur Verfügung stehen.

Die nachfolgenden Authentifizierungsoptionen sind hier zusammengefasst:

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

Verfügbare Autorisierungsoptionen

In der folgenden Tabelle werden die verfügbaren Autorisierungsoptionen aufgeführt. Für jede Option wird in der Tabelle angezeigt, ob Windows-Authentifizierung und/oder Identitätswechsel erforderlich sind. Wenn die Windows-Authentifizierung nicht erforderlich ist, steht die zugehörige Autorisierungsoption für alle anderen Authentifizierungstypen zur Verfügung. Verwenden Sie die Tabelle, um Ihre Authentifizierungs-/Autorisierungsstrategie zu optimieren.

Tabelle 8.1: *Anforderungen für die Windows-Authentifizierung und den Identitätswechsel*

Autorisierungsoption	Erfordert die Windows-Authentifizierung	Erfordert den Identitätswechsel
FileAuthorizationModule	Ja	Nein
UrlAuthorizationModule	Nein	Nein
PrincipalPermission-Forderungen	Nein	Nein
.NET-Rollen	Nein	Nein
Enterprise Services-Rollen	Ja	Ja (innerhalb der ASP.NET-Webanwendung)
NTFS-Berechtigung (für direkt angeforderte statische Dateitypen; keine Zuordnung zu einer ISAPI-Erweiterung)	N/V – Diese Dateien werden nicht von ASP.NET behandelt. Mit jedem (nicht anonymen) IIS-Authentifizierungsmechanismus sollten Berechtigungen für einzeln authentifizierte Benutzer konfiguriert werden. Bei der anonymen Authentifizierung sollten Berechtigungen für IUSR_MACHINE konfiguriert werden.	Nein (IIS führt die Zugriffsüberprüfung durch)

NTFS-Berechtigungen (für Dateien, auf die Webanwendungscode zugreift)	Nein	Nein Beim Identitätswechsel konfigurieren Sie die Zugriffssteuerungslisten für die gewechselte Windows-Identität, bei der es sich entweder um den ursprünglichen Aufrufer oder die Identität handelt, die im Element <**identity**> in der Datei **Web.config*** angegeben ist.

* Die gewechselte Identität kann der ursprüngliche Aufrufer oder die Identität sein, die im Element <**identity**> in **Web.config** angegeben ist. Betrachten Sie die folgenden zwei <**identity**>-Elemente:

```
<identity impersonate="true" />
<identity impersonate="true" userName="Bob" password="pwd" />
```

Die erste Konfiguration führt zum Identitätswechsel des ursprünglichen Aufrufers (wie durch IIS authentifiziert), während die zweite Konfiguration die Identität "Bob" ergibt. Die zweite Konfiguration wird aus zwei Gründen nicht empfohlen:

- Sie erfordert, dass Sie der ASP.NET-Prozessidentität unter dem Betriebssystem Microsoft Windows® 2000 das Recht **Als Teil des Betriebssystem handeln** gewähren.
- Weiterhin erfordert sie, dass Sie ein unverschlüsseltes Kennwort in die Datei **Web.config** einbeziehen.

Beide Einschränkungen werden in der nächsten Version des .NET Frameworks aufgehoben.

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 Anwendungsbasis für die einzelnen Anwendungen in der Datei **Web.config** konfigurieren.

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

Bei dieser Konfiguration nimmt der ASP.NET-Anwendungscode die Identität des für IIS authentifizierten Aufrufers an.

Konfigurierbare Sicherheit

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

- **Windows-Zugriffssteuerungslisten**
  - **Vom Client angeforderte Ressourcen** – Die ASP.NET-Klasse **FileAuthorizationModule** führt Zugriffsüberprüfungen für angeforderte Dateitypen durch, die der ASP.NET ISAPI zugeordnet sind. Sie verwendet das Zugriffstoken des ursprünglichen Aufrufers sowie die Zugriffssteuerungsliste, die den angeforderten Ressourcen zugeordnet ist.
    Für statische Dateitypen (keiner ISAPI-Erweiterung zugeordnet) führt IIS Zugriffsüberprüfungen durch, wobei das Zugriffstoken des Aufrufers sowie die der Datei zugeordnete Zugriffssteuerungsliste verwendet werden.
  - **Ressourcen, auf die die Anwendung zugreift –** Sie können Windows-Zugriffssteuerungslisten von Ressourcen anhand des ursprünglichen Aufrufers konfigurieren, auf die Ihre Anwendung zugreift (Dateien, Ordner, Registrierungsschlüssel, Active Directory-Objekte usw.).
- **URL-Autorisierung** – Konfigurieren Sie die URL-Autorisierung in der Datei **Web.config**. 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>
```

- **Enterprise Services (COM+)-Rollen –** Rollen werden im COM+-Katalog verwaltet. Sie können Rollen mit dem Verwaltungsprogramm oder -skript für Komponentendienste konfigurieren.

Programmatische Sicherheit

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

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

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

  - Deklarativ (Attribute, die Schnittstellen, Klassen und Methoden voranstehen)

```
[PrincipalPermission(SecurityAction.Demand,
                     Role=@"DomainName\WindowsGroup)]
```

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

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

- **Enterprise Services (COM+)-Rollen –** Sie können die Rollenüberprüfung unter Verwendung der Klasse **ContextUtil** programmatisch durchführen.

```
ContextUtil.IsCallerInRole("Manager")
```

Verwendungshinweise

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

- Die Benutzer Ihrer Anwendung verfügen über Windows-Konten, die vom Server authentifiziert werden können.
- Sie müssen den Sicherheitskontext des ursprünglichen Aufrufers in die mittlere und/oder Datenebene der Webanwendung versetzen, um eine feinstufige Autorisierung (auf Benutzerbasis) zu unterstützen.
- Sie müssen den Sicherheitskontext des ursprünglichen Aufrufers durch die nachgeschalteten Ebenen leiten, um die Überwachung auf Betriebssystemebene zu unterstützen.

Bevor Sie den Identitätswechsel in Ihrer Anwendung verwenden, stellen Sie sicher, dass Sie die relativen Vor- und Nachteile dieses Ansatzes im Vergleich zur Verwendung des vertrauenswürdigen Subsystemmodells verstehen. Diese wurden unter "Auswählen eines Modells für den Ressourcenzugriff" in Kapitel 3, "Authentifizierung und Autorisierung", erläutert.

Folgendes zählt zu den Nachteilen des Identitätswechsels:

- Verringerte Skalierbarkeit von Anwendungen aufgrund der fehlenden Möglichkeit, Datenbankverbindungen effektiv zu zusammenzufassen.
- Erhöhter Verwaltungsaufwand, da Zugriffssteuerungslisten für Back-End-Ressourcen für einzelne Benutzer konfiguriert werden müssen.
- Die Zuweisung erfordert eine Kerberos-Authentifizierung und eine entsprechend konfigurierte Umgebung.

Weitere Informationen

- Weitere Informationen zur Windows-Authentifizierung finden Sie weiter unten in diesem Kapitel unter "Windows-Authentifizierung".
- Weitere Informationen zum Identitätswechsel finden Sie weiter unten in diesem Kapitel unter "Identitätswechsel".
- Weitere Informationen zur URL-Autorisierung finden Sie weiter unten in diesem Kapitel unter "Hinweise zur URL-Autorisierung".
- Weitere Informationen zu Enterprise Services (COM+)-Rollen finden Sie in Kapitel 9, "Enterprise Services-Sicherheit".
- Weitere Informationen zu den PrincipalPermission-Forderungen finden Sie unter "Identitäten und Principals" in Kapitel 2 "Sicherheitsmodell für ASP.NET-Anwendungen".

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**
  - **Vom Client angeforderte Ressourcen** – Die ASP.NET-Klasse **FileAuthorizationModule** führt Zugriffsüberprüfungen für angeforderte Dateitypen durch, die der ASP.NET ISAPI zugeordnet sind. Sie verwendet das Zugriffstoken des ursprünglichen Aufrufers sowie die Zugriffssteuerungsliste, die den angeforderten Ressourcen zugeordnet ist. Der Identitätswechsel ist nicht erforderlich.

    Für statische Dateitypen (keiner ISAPI-Erweiterung zugeordnet) führt IIS Zugriffsüberprüfungen durch, wobei das Zugriffstoken des Aufrufers sowie die der Datei zugeordnete Zugriffssteuerungsliste verwendet werden.
  - **Ressourcen, auf die die Anwendung zugreift –** Konfigurieren Sie Windows-Zugriffssteuerungslisten von Ressourcen, auf die Ihre Anwendung zugreift (Dateien, Ordner, Registrierungsschlüssel, Active Directory-Objekte usw.), anhand der ASP.NET-Prozessidentität.
- **URL-Autorisierung** – Konfigurieren Sie die URL-Autorisierung in der Datei **Web.config**. 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 folgenden programmatischen Sicherheitsoptionen stehen zur Verfügung:

- **PrincipalPermission-Forderungen (PrincipalPermission)**
  - Imperativ

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

  - Deklarativ

```
[PrincipalPermission(SecurityAction.Demand,
                     Role=@"DomainName\WindowsGroup")]
```

- **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 Benutzer Ihrer Anwendung besitzen Windows-Konten, die vom Server authentifiziert werden können.
- Sie möchten eine feste Identität für den Zugriff auf nachgeschaltete Ressourcen verwenden (z. B. Datenbanken), um das Verbindungspooling zu unterstützen.

Weitere Informationen

- Weitere Informationen zur Windows-Authentifizierung finden Sie weiter unten in diesem Kapitel unter "Windows-Authentifizierung".
- Weitere Informationen zur URL-Autorisierung finden Sie weiter unten in diesem Kapitel unter "Hinweise zur URL-Autorisierung".
- Weitere Informationen zu den PrincipalPermission-Forderungen finden Sie unter "Sicherheitsarchitektur" im Abschnitt "ASP.NET-Autorisierungsoptionen" dieses Handbuchs.

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 Ihre Anwendung zu konfigurieren. Dies wird im folgenden Fragment der Konfigurationsdatei veranschaulicht.

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

Verwendungshinweise

Dieser Ansatz wird für die aktuelle Version (Version 1) des .NET Frameworks in sicheren Umgebungen aus zwei Gründen nicht empfohlen:

- Benutzernamen und Kennwörter sollten nicht unverschlüsselt in Konfigurationsdateien gespeichert werden, wobei dies insbesondere für Konfigurationsdateien gilt, die in virtuellen Verzeichnissen 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 Webanwendungsprozess gefährdet.

Das .NET Framework, Version 1.1, stellt für dieses Szenario unter Windows 2000 eine Erweiterung bereit:

- Die Anmeldeinformationen werden verschlüsselt.
- Die Anmeldung wird vom IIS-Prozess durchgeführt, damit ASP.NET das Recht **Als Teil des Betriebssystem handeln** nicht erfordert.

Formularauthentifizierung

Die folgenden Konfigurationselemente zeigen, wie die Formularauthentifizierung deklarativ in der Datei **Web.config** aktiviert wird.

```
<authentication mode="Forms">
  <forms loginUrl="logon.aspx" name="AuthCookie" timeout="60" path="/">
  </forms>
</authentication>
```

Konfigurierbare Sicherheit

Wenn Sie die Formularauthentifizierung verwenden, stehen Ihnen die folgenden Autorisierungsoptionen zur Verfügung:

- **Windows-Zugriffssteuerungslisten**
  - **Vom Client angeforderte Ressourcen** – Angeforderte Ressourcen erfordern Zugriffssteuerungslisten, die den Lesezugriff auf das anonyme Internetbenutzerkonto gewähren. (IIS sollte so konfiguriert sein, dass es bei der Verwendung der Formularauthentifizierung den anonymen Zugriff ermöglicht).
    Die ASP.NET-Autorisierung steht nicht zur Verfügung, da sie die Windows-Authentifizierung erfordert.
  - **Ressourcen, auf die die Anwendung zugreift** – Konfigurieren Sie Windows-Zugriffssteuerungslisten von Ressourcen, auf die Ihre Anwendung zugreift (Dateien, Ordner, Registrierungsschlüssel, Active Directory-Objekte usw.), anhand der ASP.NET-Prozessidentität.
- **URL-Autorisierung**
  Konfigurieren Sie die URL-Autorisierung in der Datei **Web.config**. Bei der Formularauthentifizierung wird das Format von Benutzernamen durch den benutzerdefinierten Datenspeicher bestimmt, bei dem es sich z. B. um eine SQL Server-Datenbank oder um Active Directory handelt.
  - Bei Verwendung eines SQL Server-Datenspeichers:

    ```
    <authorization>
    <deny users="?" />
      <allow users="Mary,Bob,Joe" roles="Manager,Sales" />
    </authorization>
    ```

  - Wenn Sie Active Directory als Datenspeicher verwenden, werden Benutzer- und Gruppennamen im X.500-Format angezeigt:

    ```
    <authorization>
      <deny users="someAccount@domain.corp.yourCompany.com" />
      <allow roles ="CN=Smith James,CN=FTE_northamerica,CN=Users,
                    DC=domain,DC=corp,DC=yourCompany,DC=com" />
    </authorization>
    ```

Programmatische Sicherheit

Die folgenden programmatischen Sicherheitsoptionen stehen zur Verfügung:

- **PrincipalPermission-Forderungen (PrincipalPermission)**
  - Imperativ

```
PrincipalPermission permCheck = new PrincipalPermission(
                                    null, "Manager");
permCheck.Demand();
```

  - Deklarativ

```
[PrincipalPermission(SecurityAction.Demand,
                     Role="Manager")]
```

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

```
IPrincipal.IsInRole("Manager");
```

Verwendungshinweise

Die Formularauthentifizierung ist am besten für Internetanwendungen geeignet. Verwenden Sie die Formularauthentifizierung für folgende Situationen:

- Die Anwendungsbenutzer verfügen über keine Windows-Konten.
- Die Benutzer sollen sich durch Eingeben von Anmeldeinformationen über ein HTML-Formular anmelden.

Weitere Informationen

- Weitere Informationen zur Formularauthentifizierung finden Sie weiter unten in diesem Kapitel unter "Formularauthentifizierung".
- Weitere Informationen zur URL-Autorisierung finden Sie weiter unten in diesem Kapitel unter "Hinweise zur URL-Autorisierung".

Passport-Authentifizierung

Die folgenden Konfigurationselemente zeigen, wie die Passport-Authentifizierung deklarativ in der Datei **Web.config** aktiviert wird.

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

Verwendungshinweise

Die Passport-Authentifizierung wird im Internet verwendet, wenn Anwendungsbenutzer über keine Windows-Konten verfügen und Sie eine Lösung mit einer einzigen Anmeldung implementieren möchten. Benutzer, die sich zuvor bei einer teilnehmenden Passport-Site mit einem Passport-Konto angemeldet haben, müssen sich bei Ihrer Website nicht anmelden, wenn diese mit der Passport-Authentifizierung konfiguriert ist.

Konfigurieren der Sicherheit

In diesem Abschnitt werden die praktischen Schritte veranschaulicht, die zum Konfigurieren der Sicherheit für eine ASP.NET-Webanwendung erforderlich sind. Diese sind in Abbildung 8.3 zusammengefasst.

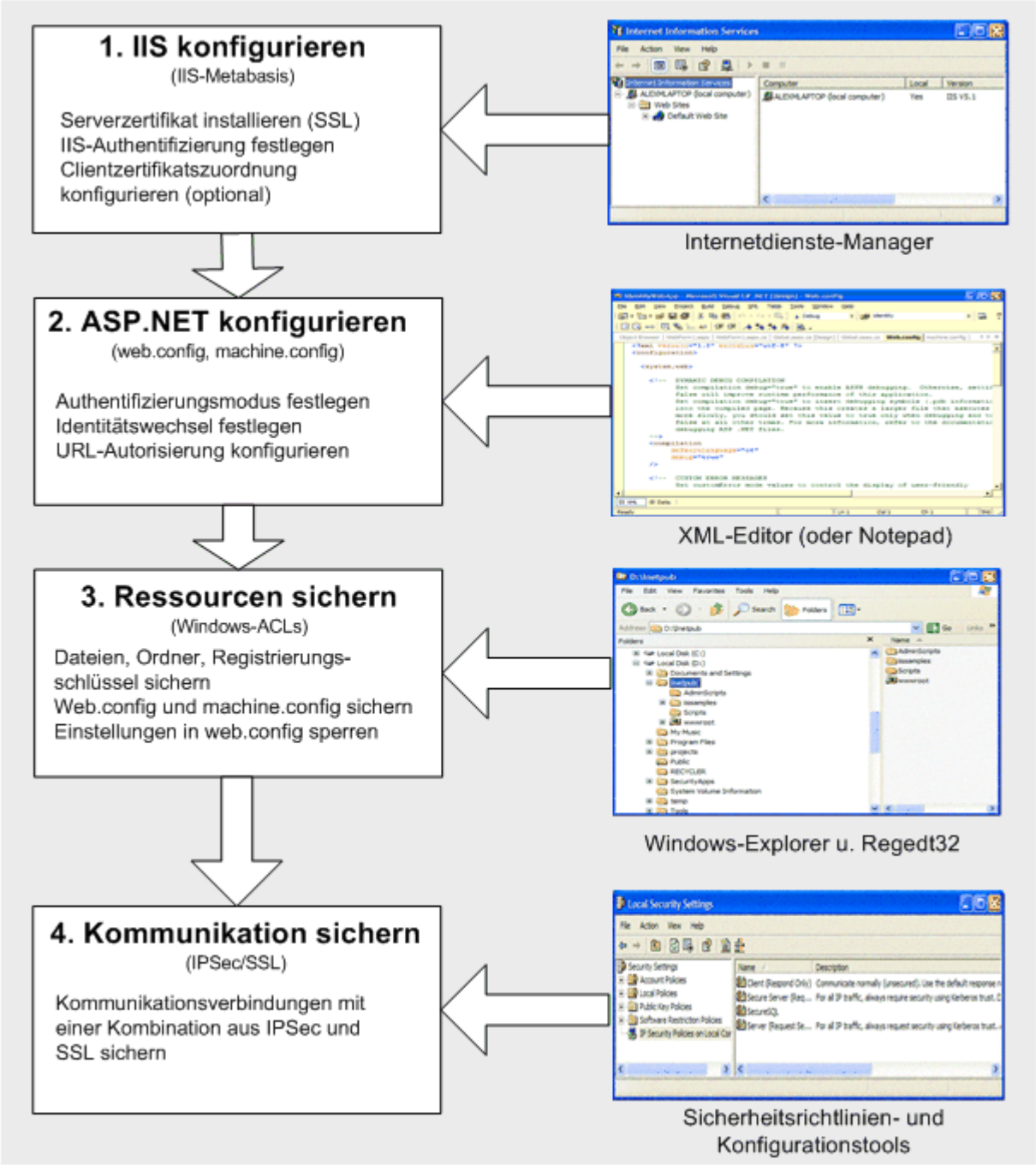

Abbildung 8.3
Konfigurieren der ASP.NET-Anwendungssicherheit

Konfigurieren der IIS-Einstellungen

Zum Konfigurieren der IIS-Sicherheit müssen Sie die folgenden Schritte ausführen:

1. Installieren Sie optional ein Webserverzertifikat (wenn SSL erforderlich ist).
   Weitere Informationen finden Sie unter "Vorgehensweise: Einrichten von SSL auf einem Webserver" im Abschnitt "Referenz" dieses Handbuchs.
2. Konfigurieren Sie die IIS-Authentifizierung.
3. Konfigurieren Sie optional die Clientzertifikatszuordnung (bei Verwendung der Zertifikatsauthentifizierung).
   Weitere Informationen zur Clientzertifikatszuordnung finden Sie in der Microsoft Knowledge Base im Artikel Q313070, "How to Configure Client Certificate Mappings in Internet Information Services (IIS) 5.0" (US).
4. Legen Sie NTFS-Berechtigungen für Dateien und Ordner fest. Zwischen diesen überprüfen IIS und **FileAuthorizationModule** von ASP.NET, dass der authentifizierte Benutzer (oder das anonyme Internetbenutzerkonto) über die Rechte für den Zugriff auf die angeforderte Datei (auf Basis der Einstellungen für die Zugriffssteuerungslisten) verfügt.

Konfigurieren der ASP.NET-Einstellungen

Die Einstellungen für die Konfiguration auf Anwendungsebene werden in **Web.config**-Dateien verwaltet, die sich im virtuellen Stammverzeichnis der Anwendung und optional in zusätzlichen Unterordnern befinden (diese Einstellungen können manchmal die Einstellungen für den übergeordneten Ordner außer Kraft setzen).

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

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

2. **Konfigurieren des Identitätswechsels –** Für ASP.NET-Anwendungen erfolgt standardgemäß kein Identitätswechsel. Die Anwendung wird unter Verwendung der konfigurierten ASP.NET-Prozessidentität (normalerweise ASPNET) ausgeführt, und sämtliche Ressourcenzugriffe dieser Anwendung werden mit dieser Identität durchgeführt. Ein Identitätswechsel ist nur unter den folgenden Bedingungen erforderlich:
   - Sie verwenden Enterprise Services und möchten Enterprise Services (COM+)-Rollen verwenden, um den Zugriff auf Funktionen zu autorisieren, die von Serviced Components bereitgestellt werden.
   - IIS wurde für die anonyme Authentifizierung konfiguriert, und Sie möchten das anonyme Internetbenutzerkonto für Ressourcenzugriffe verwenden. Weitere Informationen zu diesem Ansatz finden Sie unter "Zugreifen auf Netzwerkressourcen" weiter unten in diesem Kapitel.
   - Sie müssen den Sicherheitskontext des authentifizierten Benutzers auf die nächste Ebene versetzen (z. B. die Datenbank).
   - Sie haben eine klassische ASP-Anwendung zu ASP.NET portiert und möchten beim Identitätswechsel dasselbe Verhalten erreichen. Das klassische ASP wechselt standardmäßig die Identität des Aufrufers.

   Verwenden Sie das folgende **<identity**>-Element in der Datei **Web.config** Ihrer Anwendung, um den ASP.NET-Identitätswechsel zu konfigurieren.

   ```
   <identity impersonate="true" />
   ```

3. **Konfigurieren der Autorisierung** – Die URL-Autorisierung bestimmt, ob ein Benutzer oder eine Rolle bestimmte HTTP-Begriffe (z. B. GET, HEAD und POST) für eine bestimmte Datei verwenden kann. So implementieren Sie die URL-Autorisierung:
   a. Fügen Sie ein <**authorization**>-Element zur Datei **Web.config** hinzu, die sich im virtuellen Stammverzeichnis der Anwendung befindet.
   b. Schränken Sie den Zugriff ein, indem Sie die Attribute **allow** und **deny** verwenden. Das folgende Beispiel für die Datei **Web.config** verwendet die Windows-Authentifizierung und gewährt ausschließlich "Bob" und "Mary" den Zugriff.

```
<authorization>
  <allow users="DomainName\Bob, DomainName\Mary" />
  <deny users="*" />
</authorization>
```

Wichtig: Sie müssen am Ende des Elements <**authorization**> entweder <**deny users**="?"/> oder <**deny users**="*"/> hinzufügen, da sonst allen authentifizierten Identitäten der Zugriff gewährt wird.

Hinweise zur URL-Autorisierung

Achten Sie beim Konfigurieren der URL-Autorisierung auf Folgendes:

- "*" bezieht sich auf alle Identitäten.
- "?" bezieht sich auf nicht authentifizierte Identitäten (d. h. die anonyme Identität).
- Es ist kein Identitätswechsel erforderlich, damit die URL-Autorisierung funktioniert.
- Die Autorisierungseinstellungen in der Datei **Web.config** beziehen sich auf alle Dateien im aktuellen Verzeichnis sowie alle Unterverzeichnisse (außer ein Unterverzeichnis enthält eine eigene **Web.config**-Datei mit einem <**authorization**>-Element. In diesem Fall setzen die Einstellungen im Unterverzeichnis die Einstellungen des übergeordneten Verzeichnisses außer Kraft).

Hinweis: Die URL-Autorisierung gilt nur für Dateitypen, die von IIS zur ASP.NET ISAPI-Erweiterung, **aspnet_isapi.dll** zugeordnet sind.

Sie können das Tag <**location**> verwenden, um die Autorisierungseinstellungen einer einzelnen Datei oder einem einzelnen Verzeichnis zuzuordnen. Das folgende Beispiel veranschaulicht, wie Sie die Autorisierung auf eine bestimmte Datei anwenden (**Page.aspx**).

```
<location path="page.aspx" />
  <authorization>
    <allow users="DomainName\Bob, DomainName\Mary" />
    <deny users="*" />
  </authorization>
</location>
```

- Die Benutzer und Rollen für die URL-Autorisierung werden von Ihren Authentifizierungseinstellungen bestimmt:
  - Wenn <**authentication mode**="Windows" /> festgelegt ist, wird der Zugriff für Windows-Benutzer- und -Gruppenkonten gewährt.
    Benutzernamen besitzen die Form **Domänenname\Windows-Benutzername**.
    Rollennamen besitzen die Form **Domänenname\Windows-Gruppenname**.

Hinweis: Auf die Gruppe der lokalen Administratoren wird über **VORDEFINIERT\Administratoren** Bezug genommen. Auf die Gruppe der lokalen Benutzer wird über **VORDEFINIERT\Benutzer** Bezug genommen.

- Wenn <**authentication mode**="Forms" /> festgelegt ist, erfolgt die Autorisierung für den Benutzer und die Rollen des Objekts **IPrincipal**, das im aktuellen HTTP-Kontext gespeichert wurde. Wenn Sie z. B. Formulare für die Authentifizierung der Benutzer für eine Datenbank verwendet haben, erfolgt die Autorisierung für die aus der Datenbank abgerufenen Rollen.
- Wenn Sie <**authentication mode**="Passport" /> festgelegt haben, erfolgt die Autorisierung für die Passport-Benutzer-ID (PUID) oder für Rollen, die aus einem Datenspeicher abgerufen wurden. Sie können z. B. eine PUID einem bestimmten Konto und zu Rollengruppen zuordnen, die in einer SQL Server-Datenbank oder in Active Directory gespeichert werden.

Hinweis: Diese Funktionalität wird in das Betriebssystem Microsoft Windows Server 2003 integriert.

- Wenn Sie <**authentication mode**="None" /> festlegen, führen Sie möglicherweise keine Autorisierung durch. **None** gibt an, dass Sie keine Authentifizierung durchführen oder keines der .NET-Authentifizierungsmodule verwenden möchten, sondern Ihren eigenen benutzerdefinierten Mechanismus.
  Wenn Sie jedoch die benutzerdefinierte Authentifizierung verwenden, sollten Sie ein **IPrincipal**-Objekt mit Rollen erstellen und es in **HttpContext.User** speichern. Wenn Sie anschließend eine URL-Autorisierung durchführen, wird diese mit dem Benutzer und den Rollen ausgeführt (unabhängig davon, wie diese abgerufen werden), die im Objekt **IPrincipal** verwaltet werden.

Beispiele für die URL-Autorisierung

Die folgende Auflistung zeigt die Syntax für einige typische URL-Autorisierungsbeispiele:

- Verweigern des Zugriffs auf das anonyme Konto

```
<deny users="?" />
```

- Verweigern des Zugriffs für alle Benutzer

```
<deny users="*"/>
```

- Verweigern des Zugriffs für die Manager-Rolle

```
<deny roles="Manager"/>
```

- Beispiel für die Formularauthentifizierung

```
<configuration>
  <system.web>
      <authentication mode="Forms">
        <forms name=".ASPXUSERDEMO"
               loginUrl="login.aspx"
               protection="All" timeout="60" />
      </authentication>
      <authorization>
        <deny users="jdoe@somewhere.com" />
        <deny users="?" />
      </authorization>
  </system.web>
</configuration>
```

Weitere Informationen

Das Element <**authorization**> arbeitet mit dem aktuellen **IPrincipal**-Objekt, das in **HttpContext.User** und auch in **HttpContext.Request.RequestType** gespeichert wird.

Sichere Ressourcen

1. Verwenden Sie Windows-Zugriffssteuerungslisten, um Ressourcen zu sichern, die Dateien, Ordner und Registrierungsschlüssel enthalten.

   Wenn Sie keinen Identitätswechsel durchführen, muss jede Ressource, auf die die Anwendung zugreifen muss, eine Zugriffssteuerungsliste besitzen, die zumindest den Lesezugriff auf das ASP.NET-Prozesskonto gewährt.

   Wenn Sie einen Identitätswechsel vornehmen, müssen die Dateien und Registrierungsschlüssel über eine Zugriffssteuerungsliste verfügen, die zumindest den Lesezugriff für den authentifizierten Benutzer gewährt (oder das anonyme Internetbenutzerkonto, wenn die anonyme Authentifizierung aktiv ist).
2. Sichern Sie die Dateien **Web.config** und **Machine.config**:
   - **Verwenden Sie die richtigen Zugriffssteuerungslisten**. Wenn ASP.NET einen Identitätswechsel durchführt, erfordert die gewechselte Identität den Lesezugriff. Andernfalls erfordert die ASP.NET-Prozessidentität den Lesezugriff. Verwenden Sie die folgende Zugriffssteuerungsliste für **Web.config** und **Machine.config**:

     System: Vollzugriff

     Administratoren: Vollzugriff

     Prozessidentität oder gewechselte Identität: Lesezugriff

     Wenn Sie für das anonyme Internetbenutzerkonto (IUSR_MACHINE) keinen Identitätswechsel durchführen, sollten Sie den Zugriff auf dieses Konto verweigern.

     Hinweis: Wenn die Anwendung einer UNC-Freigabe zugeordnet ist, erfordert die UNC-Identität auch den Lesezugriff auf die Konfigurationsdateien.

- Entfernen Sie unerwünschte HTTP-Module: Die Datei **Machine.config** enthält eine Reihe von HTTP-Standardmodulen (im Element <**httpModules**>). Dabei handelt es sich um die Folgenden:
  - **WindowsAuthenticationModule**
  - **FormsAuthenticationModule**
  - **PassportAuthenticationModule**
  - **UrlAuthorizationModule**
  - **FileAuthorizationModule**
  - **OutputCacheModule**
  - **SessionStateModule**

  Wenn die Anwendung kein bestimmtes Modul verwendet, entfernen Sie dieses, um mögliche zukünftige Sicherheitsprobleme zu vermeiden, die mit einem bestimmten Modul verbunden sein können, wenn dieses in der Anwendung eingesetzt wird.

3. Optional können Sie die Konfigurationseinstellungen mithilfe des Elements <**location**> und des **allowOverride="false"**-Attributs sperren, wie nachfolgend beschrieben wird.

Sperren von Konfigurationseinstellungen

Konfigurationseinstellungen sind hierarchisch aufgebaut. Die Einstellungen der Datei **Web.config** in Unterverzeichnissen setzen die Einstellungen der Datei **Web.config** in übergeordneten Verzeichnissen außer Kraft. Die Einstellungen von **Web.config** setzen ebenfalls die Einstellungen von **Machine.config** außer Kraft.

Sie können Konfigurationseinstellungen sperren, um zu verhindern, dass diese in untergeordneten Ebenen außer Kraft gesetzt werden, indem Sie das Element <**location**> gemeinsam mit dem **allowOverride-Attribut** verwenden. Beispiel:

```
<location path="somepath" allowOverride="false" />
 . . . arbitrary configuration settings . . .
</location>
```

Beachten Sie, dass der Pfad auf eine Website oder ein virtuelles Verzeichnis verweisen kann und auf das benannte Verzeichnis sowie alle Unterverzeichnisse angewendet wird. Wenn Sie für **allowOverride** den Wert **false** festlegen, verhindern Sie, dass die im Element <**location**> angegebenen Einstellungen von einer Konfigurationsdatei außer Kraft gesetzt werden, die sich in einem untergeordneten Verzeichnis befindet. Die Möglichkeit, die Konfigurationseinstellungen zu sperren, besteht für alle Arten von Einstellungen und nicht nur für Sicherheitseinstellungen wie Authentifizierungsmodi.

Verhindern von Dateidownloads

Sie können die Klasse **HttpForbiddenHandler** verwenden, damit bestimmte Dateitypen nicht über das Web gedownloadet werden können. Diese Klasse wird intern von ASP.NET verwendet, um den Download bestimmter Systemebenendateien zu verhindern (z. B. von Konfigurationsdateien, die **Web.config** einbeziehen). Eine vollständige Liste der Dateitypen, die auf diese Weise eingeschränkt werden, finden Sie im Abschnitt <**httpHandlers**> der Datei **Machine.config**.

Sie sollten die Verwendung von **HttpForbiddenHandler** für Dateien in Betracht ziehen, die Ihre Anwendung intern verwendet und die nicht für den Download gedacht sind.

Hinweis: Sie müssen die Dateien auch mit Windows-Zugriffssteuerungslisten sichern, um zu steuern, welche Benutzer auf die Dateien zugreifen können, wenn sie auf dem Webserver angemeldet sind.

▶ **So verwenden Sie "HttpForbiddenHandler", um den Download eines bestimmten Dateityps zu verhindern:**

1. Erstellen Sie in IIS für den angegebenen Dateityp eine Anwendungszuordnung, um ihn der Datei **Aspnet_isapi.dll** zuzuordnen.
   a. Klicken Sie auf der Taskleiste auf **Start**, dann auf **Programme** und anschließend auf **Verwaltung**, um dann den Eintrag **Internet-Informationsdienste** auszuwählen.
   b. Wählen Sie das virtuelle Verzeichnis der Anwendung aus, und klicken Sie dann mit der rechten Maustaste, um die Option **Eigenschaften** auszuwählen.
   c. Wählen Sie **Anwendungseinstellungen** aus, und klicken Sie dann auf **Konfiguration**.
   d. Klicken Sie auf **Hinzufügen**, um eine neue Anwendungszuordnung zu erstellen.
   e. Klicken Sie auf **Durchsuchen**, und wählen Sie dann **c:\winnt\Microsoft.NET\Framework\v1.0.3705\aspnet_isapi.dll** aus.
   f. Geben Sie die Dateierweiterung für den am Download zu hindernden Dateityp in das Feld **Erweiterung** ein (z. B. *.abc*).
   g. Stellen Sie sicher, dass die Optionen **Alle Verben** und **Skriptmodul** aktiviert und die Option **Prüfen, ob Datei existiert** deaktiviert ist.
   h. Klicken Sie auf **OK**, um das Dialogfeld **Hinzufügen/Bearbeiten der Zuordnung von Anwendungserweiterungen** zu schließen.
   i. Klicken Sie auf **OK**, um das Dialogfeld **Anwendungskonfiguration** zu schließen und dann erneut auf **OK**, um das Dialogfeld **Eigenschaften** ebenfalls zu schließen.
2. Fügen Sie der Datei **Web.config** für den angegebenen Dateityp die Zuordnung **<HttpHandler>** hinzu.
   Nachfolgend finden Sie ein Beispiel für den Dateityp **ABC**:

```
<httpHandlers>
  <add verb="*" path="*.abc"
    type="System.Web.HttpForbiddenHandler"/>
</httpHandlers>
```

Sichere Kommunikation

Verwenden Sie eine Kombination aus SSL und IPSec (Internet Protocol Security), um die Kommunikationsverbindungen zu sichern.

Weitere Informationen

- Weitere Informationen zum Verwenden von SSL zum Sichern der Verknüpfung zum Datenbankserver finden Sie unter "Vorgehensweise: Verwenden von SSL zum Sichern der Kommunikation mit SQL Server 2000".
- Weitere Informationen zum Verwenden von IPSec zwischen zwei Computern finden Sie unter "Vorgehensweise: Verwenden von IPSec zum Sichern der Kommunikation zwischen zwei Servern".

Programmieren der Sicherheit

Nachdem Sie die konfigurierbaren Sicherheitseinstellungen der Webanwendung eingerichtet haben, müssen Sie die Autorisierungsrichtlinie der Anwendung programmatisch erweitern und optimieren. Dies umfasst das Verwenden deklarativer .NET-Attribute in den Assemblys und Durchführen imperativer Autorisierungsüberprüfungen im Code.

In diesem Abschnitt werden die hauptsächlichen Programmierschritte veranschaulicht, die zum Durchführen der Autorisierung in einer ASP.NET-Webanwendung erforderlich sind.

Autorisierungsmuster

Nachfolgend wird die Standardvorgehensweise beim Autorisieren von Benutzern in der Webanwendung zusammengefasst:

1. Abrufen von Anmeldeinformationen
2. Überprüfen von Anmeldeinformationen
3. Zuweisen von Benutzern zu Rollen
4. Erstellen eines **IPrincipal**-Objekts
5. Einfügen des **IPrincipal**-Objekts in den aktuellen HTTP-Kontext
6. Autorisieren anhand der Benutzeridentität/Rollenmitgliedschaft

---

Wichtig: Die Schritte 1 bis 5 werden von ASP.NET automatisch durchgeführt, wenn Sie die Windows-Authentifizierung konfiguriert haben. Bei den anderen Authentifizierungsmechanismen (Formularauthentifizierung, Passport-Authentifizierung und benutzerdefinierte Ansätze) müssen Sie den Code zum Durchführen dieser Schritte schreiben (wie unten beschrieben).

Abrufen von Anmeldeinformationen

Sie müssen damit beginnen, eine Reihe von Anmeldeinformationen vom Benutzer abzurufen (Benutzername und Kennwort). Wenn die Anwendung keine Windows-Authentifizierung verwendet, müssen Sie sicherstellen, dass unverschlüsselte Anmeldeinformationen im Netzwerk ordnungsgemäß unter Verwendung von SSL gesichert werden.

Überprüfen von Anmeldeinformationen

Wenn Sie die Windows-Authentifizierung konfiguriert haben, werden Anmeldeinformationen automatisch unter Verwendung der zugrunde liegenden Dienste des Betriebssystems überprüft.

Wenn Sie einen alternativen Authentifizierungsmechanismus verwenden, müssen Sie den Code schreiben, um Anmeldeinformationen anhand eines Datenspeichers (z. B. eine SQL Server-Datenbank oder Active Directory) zu überprüfen.

Weitere Informationen zum sicheren Speichern von Benutzeranmeldeinformationen in einer SQL Server-Datenbank finden Sie unter "Authentifizieren von Benutzern anhand einer Datenbank" in Kapitel 12, "Datenzugriffssicherheit".

Zuweisen von Benutzern zu Rollen

Der Benutzerdatenspeicher sollte auch eine Liste der Rollen für jeden Benutzer enthalten. Sie müssen den Code zum Abrufen der Rollenliste für den überprüften Benutzer schreiben.

Erstellen eines IPrincipal-Objekts

Die Autorisierung erfolgt anhand des authentifizierten Benutzers, dessen Identitäts- und Rollenliste im Objekt **IPrincipal** verwaltet wird (das sich im Kontext der aktuellen Webanforderung befindet).

Wenn Sie die Windows-Authentifizierung konfiguriert haben, erstellt ASP.NET automatisch ein **WindowsPrincipal**-Objekt. Dieses enthält die Identität des authentifizierten Benutzers zusammen mit einer Rollenliste, die der Liste der Windows-Gruppen entspricht, denen der Benutzer angehört.

Wenn Sie die Formular-, Passport- oder benutzerdefinierte Authentifizierung verwenden, müssen Sie in der Datei **Global.asax** im Ereignishandler **Application_AuthenticateRequest** Code hinzufügen, um das Objekt **IPrincipal** zu erstellen. Die Klasse **GenericPrincipal** wird vom .NET Framework bereitgestellt und sollte in den meisten Szenarien verwendet werden.

Einfügen des IPrincipal-Objekts in den aktuellen HTTP-Kontext

Ordnen Sie das Objekt **IPrincipal** zum aktuellen HTTP-Kontext zu (unter Verwendung der Variablen **HttpContext.User**). ASP.NET führt dies automatisch durch, wenn Sie die Windows-Authentifizierung verwenden. Andernfalls müssen Sie das Objekt manuell zuordnen.

Autorisieren anhand der Benutzeridentität und/oder Rollenmitgliedschaft

Verwenden Sie die .NET-Rollen entweder deklarativ (um die Autorisierung auf Klassen- oder Methodenebene zu erhalten) oder imperativ im Code, wenn die Anwendung eine detailliertere Autorisierungslogik erfordert.

Sie können deklarative oder imperative PrincipalPermission-Forderungen verwenden (mithilfe der Klasse **PrincipalPermission**) oder explizite Rollenüberprüfungen durchführen, indem Sie die Methode **IPrincipal.IsInRole()** aufrufen.

Im folgenden Beispiel wird bei angenommener Windows-Authentifizierung eine deklarative PrincipalPermission-Forderung veranschaulicht. Die auf das Attribut folgende Methode wird nur ausgeführt, wenn der authentifizierte Benutzer ein Mitglied der Windows-Gruppe **Manager** ist. Wenn der Aufrufer kein Mitglied dieser Gruppe ist, wird die Ausnahmebedingung **SecurityException** ausgelöst.

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

Im folgenden Beispiel wird eine explizite Rollenüberprüfung innerhalb des Codes gezeigt. Dieses Beispiel geht von der Verwendung der Windows-Authentifizierung aus. Wenn die Windows-Authentifizierung nicht verwendet wird, ist der Code jedoch sehr ähnlich. Anstatt für das Objekt **User** eine Typumwandlung in das Objekt **WindowsPrincipal** durchzuführen, sollte es in ein **GenericPrincipal**-Objekt umgewandelt werden.

```
// Extract the authenticated user from the current HTTP context.
// The User variable is equivalent to HttpContext.Current.User if you are using // an
.aspx or .asmx page
WindowsPrincipal authenticatedUser = User as WindowsPrincipal;
if (null != authenticatedUser)
{
  // Note: To retrieve the authenticated user's username, use the
  // following line of code
  // string username = authenticatedUser.Identity.Name;

  // Perform a role check
  if (authenticatedUser.IsInRole(@"DomainName\Manager") )
  {
    // User is authorized to perform manager functionality
  }
```

```
}
else
{
  // User is not authorized to perform manager functionality
}
```

Weitere Informationen

- Eine praktische Implementierung der o. a. Muster für die Formularauthentifizierung finden Sie weiter unten in diesem Kapitel unter "Formularauthentifizierung".

Erstellen einer benutzerdefinierten IPrincipal-Klasse

Die vom .NET Framework bereitgestellte Klasse **GenericPrincipal** sollte in den meisten Fällen verwendet werden, wenn Sie einen anderen Mechanismus als die Windows-Authentifizierung verwenden. Diese bietet die Rollenüberprüfung mithilfe der Methode **IPrincipal.IsInRole**.

Gelegentlich müssen Sie eine eigene **IPrincipal**-Klasse erstellen. Die Gründe zum Implementieren einer eigenen **IPrincipal**-Klasse umfassen:

- Sie möchten erweiterte Funktionen für die Rollenüberprüfung verwenden. Sie wünschen sich möglicherweise Methoden, die es Ihnen ermöglichen, zu überprüfen, ob ein bestimmter Benutzer Mitglied mehrerer Rollen ist. Beispiel:

  ```
  CustomPrincipal.IsInAllRoles( "Role", "Role2", "Role3" )
  CustomPrincipal.IsInAnyRole( "Role1", "Role2", "Role3" )
  ```

- Sie möchten eine zusätzliche Methode oder Eigenschaft implementieren, die eine Liste mit Rollen in einem Array zurückgibt. Beispiel:

  ```
  string[] roles = CustomPrincipal.Roles;
  ```

- Sie möchten für die Anwendung eine Rollenhierarchielogik durchsetzen. Ein "Senior Manager" (Geschäftsleiter) kann in der Hierarchie z. B. höher eingestuft werden als ein "Manager" (Abteilungsleiter). Dies könnte über Methoden wie die unten dargestellte getestet werden.

  ```
  CustomPrincipal.IsInHigherRole("Manager");
  CustomPrincipal.IsInLowerRole("Manager");
  ```

- Sie möchten eine verzögerte Initialisierung der Rollenlisten implementieren. Sie könnten z. B. die Rollenliste nur dann dynamisch laden, wenn eine Rollenüberprüfung angefordert wurde.
- Sie möchten die Schnittstelle **IIdentity** implementieren, damit der Benutzer durch ein **X509ClientCertificate** identifiziert wird. Beispiel:

  ```
  CustomIdentity id = CustomPrincipal.Identity;
  X509ClientCertificate cert = id.ClientCertificate;
  ```

Weitere Informationen

Weitere Informationen zum Erstellen einer eigenen **IPrincipal**-Klasse finden Sie unter "Vorgehensweise: Implementieren von IPrincipal" im Abschnitt "Referenz" dieses Handbuchs.

Windows-Authentifizierung

Verwenden Sie die Windows-Authentifizierung, wenn die Benutzer der Anwendung über Windows-Konten verfügen, die vom Server authentifiziert werden können (z. B. in Intranetszenarien).

Wenn Sie ASP.NET für die Windows-Authentifizierung konfigurieren, führt IIS die Benutzerauthentifizierung mit dem konfigurierten IIS-Authentifizierungsmechanismus durch. Dies wird in Abbildung 8.4 dargestellt.

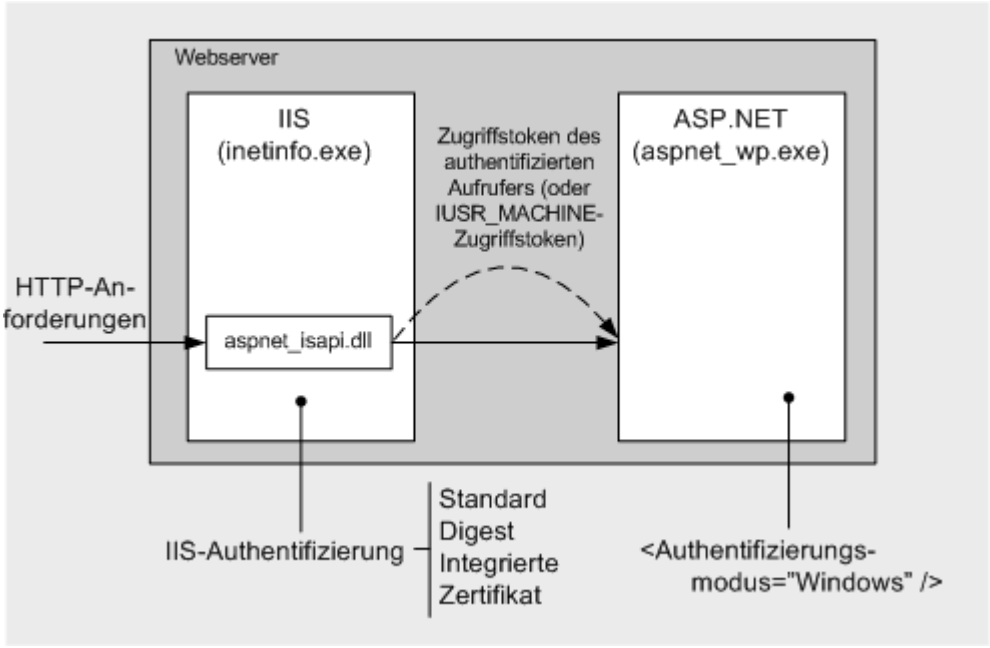

Abbildung 8.4
ASP.NET Windows-Authentifizierung verwendet IIS zum Authentifizieren von Aufrufern

Das Zugriffstoken des authentifizierten Aufrufers (bei dem es sich um das anonyme Internetbenutzerkonto handeln kann, wenn IIS für die anonyme Authentifizierung konfiguriert ist) wird für die ASP.NET-Anwendung zur Verfügung gestellt. Beachten Sie Folgendes:

- Dadurch wird es dem ASP.NET **FileAuthorizationModule** ermöglicht, Zugriffsüberprüfungen anhand der angeforderten ASP.NET-Dateien durchzuführen, die das Zugriffstoken des ursprünglichen Aufrufers verwenden.

  Wichtig: Die ASP.NET-Dateiautorisierung führt nur Zugriffsüberprüfungen für Dateitypen durch, die der Datei **Aspnet_isapi.dll** zugeordnet sind.

- Die Dateiautorisierung erfordert keinen Identitätswechsel. Bei aktiviertem Identitätswechsel verwendet jeder von der Anwendung durchgeführte Ressourcenzugriff die Identität des Aufrufers nach dem Identitätswechsel. In diesem Fall stellen Sie sicher, dass die den Ressourcen zugeordneten Zugriffssteuerungslisten einen Zugriffssteuerungseintrag (ACE, Access Control Entry) enthalten, der zumindest den Lesezugriff für die Identität des ursprünglichen Aufrufers enthält.

Identifizieren des authentifizierten Benutzers

ASP.NET ordnet der aktuellen Webanforderung ein **WindowsPrincipal**-Objekt zu. Dieses enthält die Identität des authentifizierten Windows-Benutzers zusammen mit einer Liste der Rollen, denen dieser Benutzer angehört. Bei der Windows-Authentifizierung besteht die Rollenliste aus einer Reihe von Windows-Gruppen, denen der Benutzer angehört.

Im folgenden Code wird veranschaulicht, wie die Identität des authentifizierten Windows-Benutzers abgerufen und ein einfacher Rollentest für die Autorisierung durchgeführt wird.

```
WindowsPrincipal user = User as WindowsPrincipal;
if (null != user)
{
  string username = user.Identity.Name;
  // Perform a role check
  if ( user.IsInRole(@"DomainName\Manager") )
  {
    // User is authorized to perform manager functionality
  }
}
else
{
  // Throw security exception as we don't have a WindowsPrincipal
}
```

Formularauthentifizierung

In Abbildung 8.5 wird die Reihenfolge der bei der Formularauthentifizierung von einem nicht authentifizierten Benutzer ausgelösten Ereignisse angezeigt, der versucht, auf eine gesicherte Datei oder Ressource zuzugreifen (wobei die URL-Autorisierung den Benutzerzugriff ablehnt).

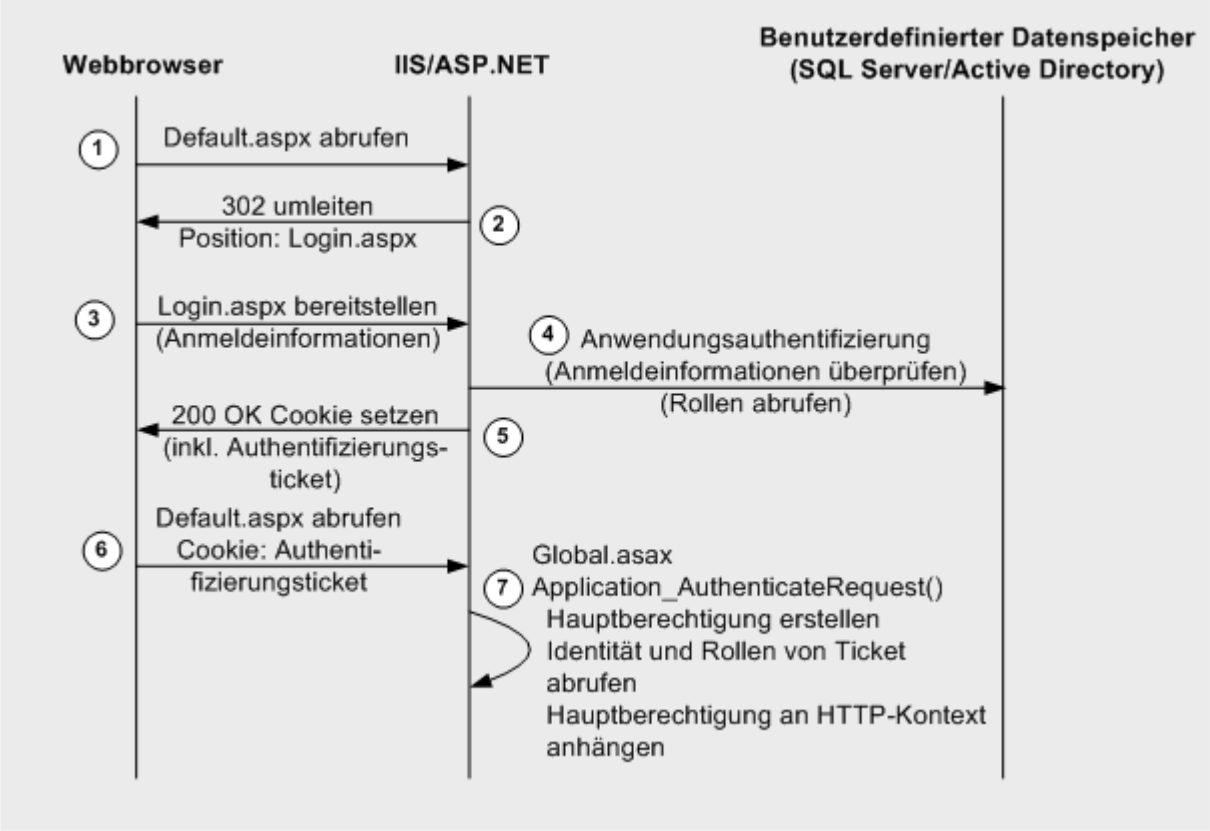

Abbildung 8.5
Ereignisabfolge bei der Formularauthentifizierung

Nachfolgend wird die in Abbildung 8.5 dargestellte Ereignisabfolge beschrieben:

1. Der Benutzer startet eine Webanforderung für die Datei **Default.aspx**.

   IIS gewährt die Anforderung, da der anonyme Zugriff aktiviert ist. ASP.NET überprüft die <**authorization**>-Elemente und findet ein <**deny users**="?" />-Element.
2. Der Benutzer wird, wie im **LoginUrl**–Attribut des Elements <**forms**> angegeben, zur Anmeldeseite (**Login.aspx**) weitergeleitet.
3. Der Benutzer stellt die Anmeldeinformationen bereit und übermittelt das Anmeldeformular.
4. Die Anmeldeinformationen werden anhand eines Speichers überprüft (SQL Server oder Active Directory), und optional werden Rollen abgerufen. Sie müssen eine Rollenliste abrufen, wenn Sie die rollenbasierte Autorisierung verwenden möchten.
5. Mit **FormsAuthenticationTicket** wird ein Cookie erstellt und an den Client zurückgesendet. Die Rollen werden optional im Ticket gespeichert. Durch das Speichern der Rollenliste im Ticket vermeiden Sie, dass Sie auf die Datenbank zugreifen müssen, um die Liste für nachfolgende Webanforderungen desselben Benutzers erneut abzurufen.
6. Der Benutzer wird mithilfe der clientseitigen Umleitung zur ursprünglich angeforderten Seite (**Default.aspx**) umgeleitet.
7. Im Ereignishandler **Application_AuthenticateRequest** (in **Global.asax**) wird das Ticket zum Erstellen eines **IPrincipal**-Objekts verwendet und in **HttpContext.User** gespeichert.

   ASP.NET überprüft die <**authorization**>-Elemente und findet ein <**deny users**="?" />-Element. Diesmal ist der Benutzer jedoch authentifiziert.

   ASP.NET überprüft die <**authorization**>-Elemente, um sicherzustellen, dass sich der Benutzer im Element <**allow**> befindet.

   Dem Benutzer wird der Zugriff auf die Datei **Default.aspx** erteilt.

Entwicklungsschritte für die Formularauthentifizierung

In der folgenden Liste werden die wichtigsten Schritte hervorgehoben, die Sie zum Implementieren der Formularauthentifizierung durchführen müssen:

1. Konfigurieren von IIS für den anonymen Zugriff.
2. Konfigurieren von ASP.NET für die Formularauthentifizierung.
3. Erstellen eines Webformulars zum Anmelden und Überprüfen der bereitgestellten Anmeldeinformationen.
4. Abrufen einer Rollenliste aus dem benutzerdefinierten Datenspeicher.
5. Erstellen eines Tickets für die Formularauthentifizierung (Rollen im Ticket speichern).
6. Erstellen eines **IPrincipal**-Objekts.
7. Einfügen des **IPrincipal**-Objekts in den aktuellen HTTP-Kontext.
8. Autorisieren des Benutzers auf Basis des/der Benutzernamen/Rollenmitgliedschaft.

Konfigurieren von IIS für den anonymen Zugriff

Das virtuelle Verzeichnis der Anwendung muss in IIS für den anonymen Zugriff konfiguriert sein.

▶ **So konfigurieren Sie IIS für den anonymen Zugriff:**

1. Starten Sie das Verwaltungsprogramm **Internet-Informationsdienste**.
2. Wählen Sie das virtuelle Verzeichnis der Anwendung aus, klicken Sie dann mit der rechten Maustaste, um die Option **Eigenschaften** auszuwählen.
3. Klicken Sie auf **Verzeichnissicherheit**.
4. Klicken Sie in der Gruppe **Authentifizierung und Zugriffssteuerung** auf **Bearbeiten**.
5. Wählen Sie **Anonymer Zugriff**.

Konfigurieren von ASP.NET für die Formularauthentifizierung

Nachfolgend wird eine Beispielkonfiguration angezeigt.

```
<authentication mode="Forms">
  <forms name="MyAppFormsAuth"
       loginUrl="login.aspx"
       protection="Encryption"
       timeout="20"
       path="/" >
  </forms>
</authentication>
```

Erstellen eines Webformulars für die Anmeldung und Überprüfen der bereitgestellten Anmeldeinformationen

Überprüfen Sie die Anmeldeinformationen anhand einer SQL Server-Datenbank oder mithilfe von Active Directory.

Weitere Informationen

- Weitere Informationen finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit SQL Server 2000" im Abschnitt "Referenz" dieses Handbuchs.
- Weitere Informationen finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit Active Directory" im Abschnitt "Referenz" dieses Handbuchs.

Abrufen einer Rollenliste aus dem benutzerdefinierten Datenspeicher

Rufen Sie Rollen aus einer Tabelle der SQL Server-Datenbank oder in Active Directory konfigurierte Gruppen-/Verteilerlisten ab. Weitere Informationen finden Sie in den zuvor aufgeführten Ressourcen.

Erstellen eines Tickets für die Formularauthentifizierung

Speichern Sie die abgerufenen Rollen im Ticket. Dies wird im nachfolgenden Code veranschaulicht.

```
// This event handler executes when the user clicks the Logon button
// having supplied a set of credentials
private void Logon_Click(object sender, System.EventArgs e)
{
  // Validate credentials against either a SQL Server database
  // or Active Directory
  bool isAuthenticated = IsAuthenticated( txtUserName.Text,
                                          txtPassword.Text );
  if (isAuthenticated == true )
  {
    // Retrieve the set of roles for this user from the SQL Server
    // database or Active Directory. The roles are returned as a
    // string that contains pipe separated role names
    // for example "Manager|Employee|Sales|"
    // This makes it easy to store them in the authentication ticket

    string roles = RetrieveRoles( txtUserName.Text, txtPassword.Text );
```

```
    // Create the authentication ticket and store the roles in the
    // custom UserData property of the authentication ticket
    FormsAuthenticationTicket authTicket = new
       FormsAuthenticationTicket(
                   1,                          // version
                   txtUserName.Text,           // user name
                   DateTime.Now,               // creation
                   DateTime.Now.AddMinutes(20),// Expiration
                   false,                      // Persistent
                   roles );                    // User data
     // Encrypt the ticket.
     string encryptedTicket = FormsAuthentication.Encrypt(authTicket);
     // Create a cookie and add the encrypted ticket to the
     // cookie as data.
     HttpCookie authCookie =
               new HttpCookie(FormsAuthentication.FormsCookieName,
                              encryptedTicket);

     // Add the cookie to the outgoing cookies collection.
     Response.Cookies.Add(authCookie);
     // Redirect the user to the originally requested page
     Response.Redirect( FormsAuthentication.GetRedirectUrl(
                        txtUserName.Text,
                        false ));
  }
}
```

Erstellen eines IPrincipal-Objekts

Erstellen Sie das **IPrincipal**-Objekt im Ereignishandler **Application_AuthenticationRequest** der Datei **Global.asax**. Verwenden Sie die Klasse **GenericPrincipal**, wenn Sie nicht die erweiterten rollenbasierten Funktionen verwenden möchten. In diesem Fall erstellen Sie eine benutzerdefinierte Klasse, die **IPrincipal** implementiert.

Einfügen des IPrincipal-Objekts in den aktuellen HTTP-Kontext

Nachfolgend wird die Erstellung des Objekts **GenericPrincipal** veranschaulicht.

```
protected void Application_AuthenticateRequest(Object sender, EventArgs e)
{
  // Extract the forms authentication cookie
  string cookieName = FormsAuthentication.FormsCookieName;
  HttpCookie authCookie = Context.Request.Cookies[cookieName];
  if(null == authCookie)
  {
    // There is no authentication cookie.
    return;
  }
  FormsAuthenticationTicket authTicket = null;
  try
  {
    authTicket = FormsAuthentication.Decrypt(authCookie.Value);
  }
  catch(Exception ex)
  {
```

```
    // Log exception details (omitted for simplicity)
    return;
  }
  if (null == authTicket)
  {
    // Cookie failed to decrypt.
    return;
  }
  // When the ticket was created, the UserData property was assigned a
  // pipe delimited string of role names.
  string[] roles = authTicket.UserData.Split(new char[]{'|'});

  // Create an Identity object
  FormsIdentity id = new FormsIdentity( authTicket );
  // This principal will flow throughout the request.
  GenericPrincipal principal = new GenericPrincipal(id, roles);
  // Attach the new principal object to the current HttpContext object
  Context.User = principal;
}
```

Autorisieren des Benutzers auf Basis des/der Benutzernamen/Rollenmitgliedschaft

Verwenden Sie deklarative PrincipalPermission-Forderungen, um den Zugriff auf Methoden einzuschränken. Verwenden Sie die imperativen PrincipalPermission-Forderungen und/oder expliziten Rollenüberprüfungen (**IPrincipal.IsInRole**), um innerhalb der Methoden eine detailliertere Autorisierung durchzuführen.

Richtlinien für die Formularimplementierung

- Verwenden Sie SSL beim Erfassen der Anmeldeinformationen mit einem HTML-Formular.

  Zusätzlich zur Verwendung von SSL für die Anmeldeseite sollten Sie SSL auch für andere Seiten verwenden, wenn die Anmeldeinformationen oder Cookies für die Authentifizierung über das Netzwerk gesendet werden. Dadurch wird die Gefahr verringert, die mit Angriffen verbunden ist, bei denen Cookies wiedergegeben werden.
- Authentifizieren Sie Benutzer anhand eines benutzerdefinierten Datenspeichers. Verwenden Sie SQL Server oder Active Directory.
- Rufen Sie eine Rollenliste aus dem benutzerdefinierten Datenspeicher ab, und speichern Sie eine Liste mit durch Trennzeichen getrennten Rollen in der Eigenschaft **UserData** der Klasse **FormsAuthenticationTicket**. Dadurch wird die Leistung durch Beseitigen des wiederholten Zugriffs auf den Datenspeicher für jede Webanforderung verbessert, und Sie müssen die Anmeldeinformationen des Benutzers auch nicht im Authentifizierungscookie speichern.
- Wenn die Liste der Rollen sehr umfangreich ist und die Gefahr besteht, den Grenzwert für die Cookiegröße zu überschreiten, speichern Sie die Rollendetails im ASP.NET-Cacheobjekt oder der ASP.NET-Datenbank und rufen diese dann für jede nachfolgende Anforderung ab.

- Gehen Sie für jede Anforderung nach der ersten Authentifizierung wie folgt vor:
  - Rufen Sie die Rollen aus dem Ticket im Ereignishandler **Application_AuthenticateRequest** ab.
  - Erstellen Sie ein **IPrincipal**-Objekt und speichern Sie dieses im HTTP-Kontext (**HttpContext.User**). Es wird vom .NET Framework ebenfalls zum aktuellen .NET-Thread (**Thread.CurrentPrincipal**) zugeordnet.
  - Verwenden Sie die Klasse **GenericPrincipal**, außer es besteht ein bestimmter Grund dafür, eine **IPrincipal**-Implementierung zu erstellen, z. B. um erweiterte rollenbasierte Operationen zu unterstützen.
- Verwenden Sie zwei Cookies: eines für die Personalisierung und das andere für die sichere Authentifizierung und Autorisierung. Legen Sie das Personalisierungscookie als persistent fest (stellen Sie sicher, dass es keine Daten enthält, die eine Anforderung zum Durchführen einer eingeschränkten Operation gewähren, z. B. das Ablegen einer Anforderung im sicheren Teil einer Site).
- Verwenden Sie für jede Webanwendung einen separaten Cookienamen (unter Verwendung des **Forms**-Attributs des Elements <**forms**>) und Pfad. Dadurch wird sichergestellt, dass Benutzer, die für eine Anwendung authentifiziert sind, nicht auch für andere Anwendungen als authentifiziert eingestuft werden, die auf demselben Webserver verwaltet werden.
- Stellen Sie sicher, dass Cookies in den Clientbrowsern aktiviert sind. Informationen zu einem Ansatz, der die Formularauthentifizierung verwendet und keine Cookies erfordert, finden Sie weiter unten in diesem Kapitel unter "Formularauthentifizierung ohne Cookies".

Weitere Informationen

- Weitere Informationen finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit SQL Server 2000" im Abschnitt "Referenz" dieses Handbuchs.
- Weitere Informationen finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit Active Directory" im Abschnitt "Referenz" dieses Handbuchs.
- Weitere Informationen finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit GenericPrincipal-Objekten" im Abschnitt "Referenz" dieses Handbuchs.

Verwalten mehrerer Anwendungen mithilfe der Formularauthentifizierung

Wenn Sie mehrere Webanwendungen verwalten, die die Formularauthentifizierung auf demselben Webserver verwenden, ist es möglich, dass ein für eine Anwendung authentifizierter Benutzer eine Anforderung an eine andere Anwendung stellen kann, ohne zur Anmeldeseite dieser Anwendung umgeleitet zu werden. Die Regeln der URL-Autorisierung der zweiten Anwendung verweigern möglicherweise den Zugriff für den Benutzer, ohne ihm die Möglichkeit zu bieten, Anmeldeinformationen über das Anmeldeformular bereitzustellen.

Dies geschieht nur, wenn die Attribute für Name und Pfad im Element <**forms**> für mehrere Anwendungen dieselben sind und alle Anwendung ein gemeinsames <**machineKey**>-Element in der Datei **Web.config** verwenden.

Weitere Informationen

Weitere Informationen zu diesem Thema und zu Lösungsverfahren finden Sie in den folgenden Artikeln der Knowledge Base:

- Q313116, "PRB: Forms Authentication Requests Are Not Directed to loginUrl Page" (US)
- Q310415, "PRB: Mobile Forms Authentication and Different Web Applications" (US)

Formularauthentifizierung ohne Cookies

Wenn Sie eine Lösung für die Formularauthentifizierung suchen, die keine Cookies erfordert, sollten Sie die Verwendung des Ansatzes in Erwägung ziehen, der vom Microsoft Mobile Internet Toolkit verwendet wird. Die Authentifizierung über mobile Formulare baut auf der Formularauthentifizierung auf, verwendet jedoch die Abfragezeichenfolge, um anstatt eines Cookies das Authentifizierungsticket zu übermitteln.

Weitere Informationen

Weitere Informationen zur Authentifizierung über mobile Formulare finden Sie in der Microsoft Knowledge Base im Artikel Q311568, "INFO: How To Use Mobile Forms Authentication with Microsoft Mobile Internet Toolkit" (US).

Passport-Authentifizierung

Verwenden Sie die Passport-Authentifizierung, wenn die Benutzer Ihrer Anwendung über Passport-Konten verfügen und Sie eine Lösung mit einer einzigen Anmeldung für andere Sites implementieren möchten, die für Passport aktiviert sind.

Wenn Sie ASP.NET für die Passport-Authentifizierung konfigurieren, wird der Benutzer zum Anmelden aufgefordert und dann zur Passport-Site umgeleitet. Nach der erfolgreichen Überprüfung der Anmeldeinformationen wird der Benutzer wieder zu Ihrer Website zurückgeleitet.

Konfigurieren von ASP.NET für die Passport-Authentifizierung

Verwenden Sie die folgenden Einstellungen für die Datei **Web.config**, um ASP.NET für die Passport-Authentifizierung zu konfigurieren.

```
<authentication mode="Passport">
  <passport redirectUrl="internal" />
</authentication>
<authorization>
  <deny users="?" />
  <allow users="*" />
</authorization>
```

Zuordnen einer Passport-Identität zu Rollen in Global.asax

Implementieren Sie den Ereignishandler **PassportAuthentication_OnAuthentication** in der Datei **Global.asax** wie unten gezeigt, um eine Passport-Identität zu Rollen zuzuordnen.

```
void PassportAuthentication_OnAuthenticate(Object sender,
                                           PassportAuthenticationEventArgs e)
{
  if(e.Identity.Name == "0000000000000001")
  {
    string[] roles = new String[]{"Developer", "Admin", "Tester"};
    Context.User = new GenericPrincipal(e.Identity, roles);
  }
}
```

Testen der Rollenmitgliedschaft

Das folgende Codefragment zeigt, wie die authentifizierte Passport-Identität abgerufen und die Rollenmitgliedschaft in einer ASPX-Seite überprüft wird.

```
PassportIdentity passportId = Context.User.Identity as PassportIdentity;
if (null == passportId)
{
  Response.Write("Not a PassportIdentity<br>");
  return;
}
Response.Write("IsInRole: Develeoper? " + Context.User.IsInRole("Developer"));
```

Benutzerdefinierte Authentifizierung

Wenn keines der im .NET Framework bereitgestellten Authentifizierungsmodule Ihren konkreten Authentifizierungsanforderungen entspricht, können Sie die benutzerdefinierte Authentifizierung verwenden und einen eigenen Authentifizierungsmechanismus implementieren. Ihr Unternehmen könnte z. B. schon über eine benutzerdefinierte Authentifizierungsstrategie verfügen, die von anderen Anwendungen ausgiebig genutzt wird.

So implementieren Sie die benutzerdefinierte Authentifizierung in ASP.NET:

- Konfigurieren Sie den Authentifizierungsmodus in der Datei **Web.config** wie nachfolgend gezeigt. Dadurch wird ASP.NET darüber informiert, dass keine integrierten Authentifizierungsmodule aufgerufen werden sollen.

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

- Erstellen Sie eine Klasse, die die Schnittstelle **System.Web.IHttpModule** integriert, um ein benutzerdefiniertes HTTP-Modul zu erstellen. Dieses Modul sollte einen Hook im Ereignis **HttpApplication.AuthenticateRequest** implementieren und einen Delegat bereitstellen, der bei jeder Anforderung für die Anwendung aufgerufen wird, wenn die Authentifizierung erforderlich ist.

  Folgende Kriterien müssen von einem Authentifizierungsmodul erfüllt werden:

  - Abrufen der Anmeldeinformationen vom Aufrufer.
  - Überprüfen der Anmeldeinformationen anhand eines Speichers.
  - Erstellen eines **IPrincipal**-Objekts und Speichern in **HttpContext.User**.
  - Erstellen und Schützen eines Authentifizierungstokens und dieses an den Benutzer zurücksenden (normalerweise in einer Abfragezeichenfolge, in einem Cookie oder einem verborgenen Formularfeld).
  - Abrufen des Authentifizierungstokens bei nachfolgenden Anforderungen und dieses dann überprüfen und erneut ausstellen.

Weitere Informationen

Weitere Informationen zum Implementieren eines benutzerdefinierten HTTP-Moduls finden Sie in der Microsoft Knowledge Base im Artikel Q307996, "HOW TO: Create an ASP.NET HTTP Module Using Visual C# .NET" (US).

Prozessidentität für ASP.NET

Führen Sie ASP.NET (insbesondere den Workerprozess **Aspnet_wp.exe**) mithilfe eines Kontos aus, das minimale Rechte besitzt.

Verwenden eines Kontos mit minimalen Rechten

Verwenden Sie ein Konto mit minimalen Rechten, um die Bedrohung durch das Angreifen von Prozessen zu verringern. Wenn es einem entschlossenen Angreifer gelingt, den ASP.NET-Prozess anzugreifen, der Ihre Webanwendung ausführt, kann dieser die Rechte und Zugriffsrechte, die dem Prozesskonto gewährt wurden, problemlos übernehmen und ausnutzen. Ein Konto, das mit den minimalen Rechten konfiguriert wurde, schränkt den potenziell anzurichtenden Schaden ein.

Vermeiden der Ausführung als SYSTEM

Verwenden Sie zum Ausführen von ASP.NET nicht das mit vielen Rechten versehene Konto SYSTEM, und gewähren Sie dem ASP.NET-Prozesskonto nicht das Recht **Als Teil des Betriebssystem handeln**. Möglicherweise sind Sie versucht, die eine oder andere Variante zu verwenden, damit Sie es dem Code ermöglichen, die API **LogonUser** aufzurufen, um eine feste Identität zu erhalten (normalerweise für den Zugriff auf Netzwerkressourcen). Alternative Ansätze finden Sie unter "Zugreifen auf Netzwerkressourcen" weiter unten in diesem Kapitel.

Nachfolgend finden Sie Gründe, warum die Ausführung nicht als SYSTEM erfolgen oder das Recht **Als Teil des Betriebssystems handeln** nicht gewährt werden sollte:

- Der Schaden, den ein Angreifer anrichten kann, wird erheblich erhöht, jedoch nicht die Möglichkeit, das System anzugreifen.
- Das Prinzip der minimalen Rechte wird nicht befolgt. Das Konto ASPNET wurde speziell als Konto mit minimalen Rechten konfiguriert, um mit ASP.NET-Webanwendungen ausgeführt zu werden.

Weitere Informationen

Weitere Informationen zum Recht **Als Teil des Betriebssystems handeln** finden Sie im Microsoft Systems Journal, Ausgabe August 1999, in der Spalte Security Briefs (nur auf Englisch verfügbar).

Domänencontroller und das ASP.NET-Prozesskonto

Im Allgemeinen wird nicht empfohlen, den Webserver auf einem Domänencontroller auszuführen, da eine Gefährdung des Servers auch eine Gefährdung der Domäne darstellt. Wenn Sie ASP.NET auf einem Domänencontroller ausführen müssen, sollte das ASP.NET-Prozesskonto die entsprechenden Rechte erhalten, wie in der Microsoft Knowledge Base im Artikel Q315158, "BUG: ASP.NET Does Not Work with the Default ASPNET Account on a Domain Controller" (US) beschrieben wird.

Verwenden des ASPNET-Standardkontos

Das lokale ASPNET-Konto wurde speziell für die Ausführung von ASP.NET-Webanwendungen mit minimalen Rechten konfiguriert. Verwenden Sie wenn möglich immer ASPNET.

ASP.NET-Webanwendungen werden standardmäßig mit diesem Konto ausgeführt, wie in der Datei **Machine.config** durch das Element <**processModel**> konfiguriert.

```
<processModel userName="machine" password="AutoGenerate" />
```

Hinweis: Der Benutzername **machine** kennzeichnet das ASPNET-Konto. Das Konto wird beim Installieren des .NET Frameworks mit einem stark verschlüsselten Kennwort erstellt. Zusätzlich zur Konfigurierung in der SAM-Datenbank (Security Account Manager) wird das Kennwort auf dem lokalen Computer in der Local Security Authority (LSA) gespeichert. Das System ruft das Kennwort von der LSA ab, wenn der ASP.NET-Workerprozess gestartet wird.

Wenn Ihre Anwendung auf Netzwerkressourcen zugreift, muss das ASPNET-Konto vom Remotecomputer authentifiziert werden können. Ihnen bieten sich zwei Möglichkeiten:

- Setzen Sie das Kennwort des ASPNET-Kontos auf einen bekannten Wert zurück, und erstellen Sie dann ein Duplikat des Kontos (mit demselben Namen und demselben Kennwort) auf dem Remotecomputer. Dieser Ansatz stellt unter den folgenden Umständen die einzige Option dar:
  - Der Webserver und der Remotecomputer befinden sich in separaten Domänen ohne Vertrauensstellung.
  - Der Webserver und der Remotecomputer sind voneinander durch einen Firewall getrennt, und Sie möchten die erforderlichen Anschlüsse zum Unterstützen der Windows-Authentifizierung nicht öffnen.
- Wenn die einfache Verwaltung Ihr Hauptanliegen darstellt, verwenden Sie ein Domänenkonto mit minimalen Rechten.

  Sie können ein Domänenkonto mit minimalen Rechten zum Ausführen von ASP.NET verwenden, um das manuelle Aktualisieren und Synchronisieren von Kennwörtern zu vermeiden. Es ist äußerst wichtig, dass das Domänenkonto vollständig gesperrt wird, um die Bedrohung für Prozesse zu verringern. Wenn es einem Angreifer gelingt, den ASP.NET-Workerprozess anzugreifen, besitzt diese Person die Möglichkeit, auf die Domänenressourcen zuzugreifen, wenn das Konto nicht vollständig gesperrt ist.

Hinweis: Wenn Sie ein lokales Konto verwenden und dieses erfolgreich angegriffen wird, können nur die Computer weiter angegriffen werden, auf denen Sie Duplikate des Kontos erstellt haben. Wenn Sie ein Domänenkonto verwenden, ist das Konto für jeden Computer in der Domäne sichtbar. Das Konto muss jedoch weiterhin über die entsprechende Berechtigung verfügen, um auf diese Computer zugreifen zu können.

Das Element <processModel>

Das Element <**processModel**> in der Datei **Machine.config** enthält die Attribute **userName** und **password**, die das Konto festlegen, das zum Ausführen des ASP.NET-Workerprozesses (**Aspnet_wp.exe**) verwendet werden soll. Zum Konfigurieren dieser Einstellung stehen eine Reihe von Optionen zur Auswahl. Beispiel:

- **"machine"** – Der Workerprozess wird als ASPNET-Standardkonto mit minimalen Rechten ausgeführt. Das Konto verfügt über den Netzwerkzugriff, kann jedoch nicht für andere Computer im Netzwerk authentifiziert werden, da das Konto für den Computer lokal verfügbar und somit keine Autorität vorhanden ist, die sich für das Konto verbürgen kann. Im Netzwerk wird dieses Konto als **Computername\ASPNET** dargestellt.
- **"system"** – Der Workerprozess wird als lokales SYSTEM-Konto ausgeführt. Dieses Konto verfügt über umfangreiche Rechte auf dem lokalen Computer und besitzt die Möglichkeit, auf das Netzwerk mithilfe der Anmeldeinformationen des Computers zuzugreifen. Im Netzwerk wird dieses Konto als **Domänenname\Computername$** dargestellt.

- **Bestimmte Anmeldeinformationen** – Wenn Sie Anmeldeinformationen für die Attribute **userName** und **password** bereitstellen, denken Sie an das Prinzip mit den minimalen Rechten. Wenn Sie ein lokales Konto angeben, kann die Webanwendung nicht im Netzwerk authentifiziert werden, wenn Sie nicht ein Duplikat des Kontos auf dem Remotecomputer erstellen. Wenn Sie sich dafür entscheiden, ein Domänenkonto mit minimalen Rechten zu verwenden, stellen Sie sicher, dass es sich nicht um ein Konto handelt, das für den Zugriff auf mehr Computer im Netzwerk berechtigt ist, als notwendig.

  Im .NET Framework, Version 1.1, bietet sich Ihnen die Möglichkeit, die Attribute **userName** und **password** verschlüsselt in der Registrierung zu speichern.

Hinweis: Im Gegensatz zur Ausführungsweise der klassischen ASP-Anwendungen wird der ASP.NET-Code niemals im Prozess der Datei **dllhost.exe** oder als IWAM_MACHINENAME-Konto ausgeführt, wenn die Schutzebene der Anwendung in IIS den Wert **Hoch** (**isoliert)** erhält.

An IIS gesendete ASP.NET-Anforderungen werden direkt zum ASP.NET-Workerprozess umgeleitet (**Aspnet_wp.exe**). Die ASP.NET ISAPI-Erweiterung, **Aspnet_isapi.dll** wird im Prozessadressraum von IIS (**Inetinfo.exe**) ausgeführt. (Dieser wird durch den Metabasiseintrag **InProcessIsapiApps** gesteuert, der nicht geändert werden sollte.) Die ISAPI-Erweiterung ist für das Umleiten von Anforderungen an den ASP.NET-Workerprozess verantwortlich. ASP.NET-Anwendungen werden dann im ASP.NET-Workerprozess ausgeführt, wobei Anwendungsdomänen die Isolationsgrenzen bereitstellen.

In IIS 6 haben Sie die Möglichkeit, ASP.NET-Anwendungen durch Konfigurieren von Anwendungspools zu isolieren, wobei jeder Pool eine eigene Anwendungsinstanz besitzt.

Weitere Informationen

- Weitere Informationen zum Zugreifen auf Netzwerkressourcen von ASP.NET-Webanwendungen finden Sie unter "Zugreifen auf Netzwerkressourcen" weiter unten in diesem Kapitel.
- Ausführliche Informationen zum Erstellen eines benutzerdefinierten Kontos zum Ausführen von ASP.NET finden Sie unter "Vorgehensweise: Erstellen eines benutzerdefinierten Kontos zum Ausführen von ASP.NET" im Abschnitt "Referenz" dieses Handbuchs.

Identitätswechsel

Mit der Einführung von **FileAuthorizationModule** und der effizienten Verwendung von Gatekeepern und Vertrauensgrenzen kann sich der Identitätswechsel in ASP.NET eher als Nachteil denn als Vorteil gestalten.

Identitätswechsel und lokale Ressourcen

Wenn Sie den Identitätswechsel verwenden und über Ihren Webanwendungscode auf lokale Ressourcen zugreifen, müssen Sie die jeder gesicherten Ressource zugeordneten Zugriffssteuerungslisten konfigurieren, damit diese einen ACE enthalten, der dem authentifizierten Benutzer zumindest den Lesezugriff gewährt.

Einen besseren Ansatz zum Vermeiden des Identitätswechsels stellt das Gewähren von Berechtigungen für das ASP.NET-Prozesskonto und die Verwendung der URL-Autorisierung, Dateiautorisierung und einer Kombination aus deklarativen und imperativen Überprüfungen dar, die auf Rollen basieren.

Identitätswechsel und Remoteressourcen

Wenn Sie den Identitätswechsel verwenden und dann über Ihren Webanwendungscode auf Remoteressourcen zugreifen, schlägt dieser Zugriff fehl, wenn Sie nicht die Standard-, Formular- oder Kerberos-Authentifizierung verwenden. Wenn Sie die Kerberos-Authentifizierung verwenden, müssen die Benutzerkonten ordnungsgemäß für die Delegierung konfiguriert sein. Sie müssen in Active Directory als **Konto ist vertraulich und kann nicht delegiert werden** gekennzeichnet sein.

Weitere Informationen

Weitere Informationen zum Konfigurieren der Kerberos-Delegierung finden Sie unter:

- "Übermitteln des ursprünglichen Aufrufers an die Datenbank" in Kapitel 5, "Intranetsicherheit".
- "Vorgehensweise: Implementieren der Kerberos-Delegierung unter Windows 2000" im Abschnitt "Referenz" dieses Handbuchs.

Identitätswechsel und Threading

Wenn ein Thread, der einen Identitätswechsel durchgeführt hat, einen neuen Thread erstellt, erbt der neue Thread den Sicherheitskontext des ASP.NET-Prozesskontos und nicht das gewechselte Konto.

Zugreifen auf Systemressourcen

ASP.NET führt standardmäßig keinen Identitätswechsel durch. Wenn die Webanwendung daher auf lokale Systemressourcen zugreift, erfolgt dies über den Sicherheitskontext, der dem Workerprozess **Aspnet_wp.exe** zugeordnet ist. Der Sicherheitskontext wird vom Konto bestimmt, das zum Ausführen des Workerprozesses verwendet wird.

Zugreifen auf das Ereignisprotokoll

Konten mit minimalen Rechten verfügen über ausreichend Berechtigungen, um Datensätze mithilfe vorhandener Ereignisquellen in das Ereignisprotokoll zu schreiben. Sie verfügen jedoch nicht über ausreichend Berechtigungen, um neue Ereignisquellen zu erstellen. Dies erfordert einen neuen Eintrag, der unter der folgenden Registrierungsstruktur eingefügt wird.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\<log>
```

Erstellen Sie bei Verfügbarkeit von Administratorrechten die von der Anwendung bei der Installation verwendeten Ereignisquellen, um dieses Problem zu vermeiden. Ein guter Ansatz ist es, eine .NET-Installationsklasse zu verwenden, die vom Windows Installer (wenn Sie die MSI-Bereitstellung verwenden) oder vom Systemdienstprogramm **InstallUtil.exe** (wenn Sie sie nicht verwenden) instanziiert werden kann.

Wenn Sie keine Ereignisquellen bei der Installation erstellen können, müssen Sie die Berechtigung zum folgenden Registrierungsschlüssel hinzufügen und dem ASP.NET-Prozesskonto den Zugriff gewähren (eines beliebigen gewechselten Kontos, wenn Ihre Anwendung den Identitätswechsel verwendet).

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog
```

Die Konten müssen die folgenden minimalen Rechte besitzen:

- Abfragen von Schlüsselwerten
- Festlegen von Schlüsselwerten
- Erstellen von Unterschlüsseln
- Auflisten von Unterschlüsseln
- Benachrichtigen
- Lesen

Folgender Code kann zum Schreiben von ASP.NET in das Anwendungsereignisprotokoll verwendet werden, nachdem die Berechtigungen auf die Registrierung angewendet wurden:

```
string source = "Your Application Source";
string logToWriteTo = "Application";
string eventText = "Sample Event";

if (!EventLog.SourceExists(source))
{
  EventLog.CreateEventSource(source, logToWriteTo);
}
EventLog.WriteEntry(source, eventText, EventLogEntryType.Warning, 234);
```

Zugreifen auf die Registrierung

Jeder Registrierungsschlüssel, auf den Ihre Anwendung zugreift, erfordert einen ACE in der Zugriffssteuerungsliste, die (zumindest) den Lesezugriff auf das ASP.NET-Prozesskonto gewährt.

Weitere Informationen

Weitere Informationen zu Installationsklassen und dem Dienstprogramm "InstallUtil.exe" finden Sie bei den .NET Framework-Tools in MSDN.

Zugreifen auf COM-Objekte

Beim klassischen ASP werden Anforderungen mithilfe von Threads aus dem STA-Threadpool (Single Threaded Apartment) verarbeitet. Bei ASP.NET werden Anforderungen mithilfe von Threads aus dem MTA-Threadpool (Multithreaded Apartment) verarbeitet. Dies hat Auswirkungen für ASP.NET-Webanwendungen, die Apartmentmodellobjekte aufrufen.

Apartmentmodellobjekte

Wenn eine ASP.NET-Webanwendung ein Apartmentmodellobjekt aufruft (z. B. ein Visual Basic 6 COM-Objekt), sind zwei Details zu beachten:

- Sie müssen die ASP.NET-Seite durch die Direktive **AspCompat** kennzeichnen, wie nachfolgend gezeigt.

  ```
  <%@ Page Language="C#" AspCompat="True" %>
  ```

- Erstellen Sie die COM-Objekte nicht außerhalb bestimmter Seitenereignishandler. Erstellen Sie die COM-Objekte immer in Seitenereignishandlern (z. B. **Page_Load** und **Page_Init**). Erstellen Sie die COM-Objekte nicht im Konstruktor der Seite.

Erforderliche Direktive "AspCompat"

Standardmäßig verwendet ASP.NET MTA-Threads zum Verarbeiten von Anforderungen. Dies führt zu einem Threadwechsel, wenn ein Apartmentmodell von ASP.NET aufgerufen wird, da die MTA-Threads nicht direkt auf das Apartmentmodellobjekt zugreifen können (COM verwendet stattdessen einen STA-Thread).

Das Festlegen von **AspCompat** führt dazu, dass die Seite von einem STA-Thread verarbeitet wird. Dadurch wird der Threadwechsel von MTA zu STA vermieden. Dies ist aus Sicherheitsgründen von Bedeutung, wenn Ihre Webanwendung einen Identitätswechsel durchführt, da ein Threadwechsel zu einem verlorenen Token für den Identitätswechsel führt. Das Token für den Identitätswechsel wird dem neuen Thread nicht zugeordnet.

Die Direktive **AspCompat** wird für ASP.NET-Webdienste nicht unterstützt. Das bedeutet, dass beim Aufrufen von Apartmentmodellobjekten über den Webdienstcode ein Threadwechsel eintritt und Sie dann das Token für den Identitätswechsel verlieren. Dies führt normalerweise zu einer Ausnahmebedingung, bei der der Zugriff verweigert wird.

Weitere Informationen

- Weitere Informationen finden Sie in den folgenden Knowledge Base-Artikeln:
  - Artikel Q303375, "INFO: XML Web Services and Apartment Objects" (US)
  - Artikel Q325791, "PRB: Access Denied Error Message Occurs When Impersonating in ASP.NET and Calling STA COM Components" (US)
- Weitere Informationen zum Bestimmen der Identität des momentan ausgeführten Codes finden Sie unter "Wer bin ich?" in Kapitel 13, "Problembehandlung".

Keine COM-Objekte außerhalb bestimmter Seitenereignisse erstellen

Erstellen Sie keine COM-Objekte außerhalb bestimmter Seitenereignishandler. Das folgende Codefragment zeigt, was nicht durchgeführt werden sollte.

```
<%@ Page Language="C#" AspCompat="True" %>
<script runat="server">
  // COM object created outside of Page events
  YourComObject obj = new apartmentObject();
  public void Page_Load()
  {
    obj.Foo()
  }
</script>
```

Wenn Sie Apartmentmodellobjekte verwenden, ist es wichtig, das Objekt, wie nachfolgend dargestellt, innerhalb bestimmter Seitenereignisse wie **Page_Load** zu erstellen.

```
<%@ Page Language="C#" AspCompat="True" %>
<script runat="server">
public void Page_Load()
{
  YourComObject obj  = new apartmentObject();
  obj.Foo()
}
</script>
```

Weitere Informationen

Weitere Informationen finden Sie in der Microsoft Knowledge Base im Artikel Q308095, "PRB: Creating STA Components in the Constructor in ASP.NET ASPCOMPAT Mode Negatively Impacts Performance" (US).

C#- und VB .NET-Objekte in COM+

Das Entwicklungstool Microsoft C#® und das Entwicklungssystem Microsoft Visual Basic® .NET unterstützen sämtliche Threadingmodelle (Free-Thread, Neutral, Beide und Apartment). Standardmäßig werden C#- und Visual Basic .NET-Objekte als **Beide** gekennzeichnet, wenn sie in COM+ verwaltet werden. Daher erfolgt der Zugriff direkt, und Sie verursachen keinen Threadwechsel, wenn sie von ASP.NET aufgerufen werden.

Zugreifen auf Netzwerkressourcen

Ihre Anwendung muss möglicherweise auf Netzwerkressourcen zugreifen. Dabei ist es wichtig, dass Sie in der Lage sind, Folgendes zu kennzeichnen:

- Die Ressource, auf die Ihre Anwendung zugreifen muss.
  Beispielsweise Dateien auf Dateifreigaben, Datenbanken, DCOM-Server, Active Directory-Objekte usw.
- Die zum Durchführen des Ressourcenzugriffs verwendete Identität.
  Wenn Ihre Anwendung auf Remoteressourcen zugreift, muss diese Identität vom Remotecomputer authentifiziert werden können.

Hinweis: Weitere Informationen zum Zugreifen auf SQL Server-Remotedatenbanken finden Sie in Kapitel 12, "Datenzugriffssicherheit".

Sie können über eine ASP.NET-Anwendung auf Remoteressourcen zugreifen, indem Sie eines der folgenden Verfahren verwenden:

- Verwenden der ASP.NET-Prozessidentität.
- Verwenden einer Serviced Component.
- Verwenden des anonymen Internetbenutzerkontos (z. B. IUSR_MACHINE).
- Verwenden der API **LogonUser** und Durchführen eines Identitätswechsels für eine bestimmte Windows-Identität.
- Verwenden des ursprünglichen Aufrufers.

Verwenden der ASP.NET-Prozessidentität

Wenn die Anwendung nicht für den Identitätswechsel konfiguriert wurde, stellt die ASP.NET-Prozessidentität die Standardidentität bereit, wenn die Anwendung versucht, auf Remoteressourcen zuzugreifen. Wenn Sie das ASP.NET-Prozesskonto für den Zugriff auf Remoteressourcen verwenden möchten, bieten sich Ihnen drei Möglichkeiten:

- Verwenden von gespiegelten Konten.
  Die ist der einfachste Ansatz. Sie erstellen auf dem Remotecomputer ein lokales Konto mit übereinstimmendem Benutzernamen und Kennwort. Sie müssen das Kennwort des ASPNET-Kontos im Benutzer-Manager in einen bekannten Wert ändern (verwenden Sie immer ein starkes Kennwort). Sie müssen dieses dann explizit für das Element <**processModel**> in der Datei **Machine.config** ändern und den vorhandenen Wert **AutoGenerate** ersetzen.

Wichtig: Wenn Sie das ASPNET-Kennwort in einen bekannten Wert ändern, stimmt das Kennwort in der LSA nicht länger mit dem Kennwort des SAM-Kontos überein. Wenn Sie den Standardwert **AutoGenerate** zurücksetzen möchten, müssen Sie eine der folgenden Optionen durchführen:

Ausführen von **Aspnet_regiis.exe**, um ASP.NET in die Standardkonfiguration zurückzuversetzen. Weitere Informationen finden Sie in der Microsoft Knowledge Base im Artikel Q306005, "HOWTO: Repair IIS Mapping After You Remove and Reinstall IIS" (US).

- Erstellen eines benutzerdefinierten Kontos mit minimalen Rechten zum Ausführen von ASP.NET und Erstellen eines duplizierten Kontos auf dem Remotecomputer.
- Ausführen von ASP.NET mithilfe eines Domänenkontos mit minimalen Rechten.
  Dies setzt voraus, dass sich die Client- und Servercomputer in derselben oder in vertrauten Domänen befinden.

Weitere Informationen

Weitere Informationen zum Konfigurieren eines ASP.NET-Prozesskontos finden Sie unter "Vorgehensweise: Erstellen eines benutzerdefinierten Kontos zum Ausführen von ASP.NET" im Abschnitt "Referenz" dieses Handbuchs.

Verwenden einer Serviced Component

Sie können eine prozessexterne Serviced Component für den Zugriff auf Netzwerkressourcen verwenden, die für die Ausführung als feste Identität konfiguriert ist. Diese Vorgehensweise ist in Abbildung 8.6 dargestellt.

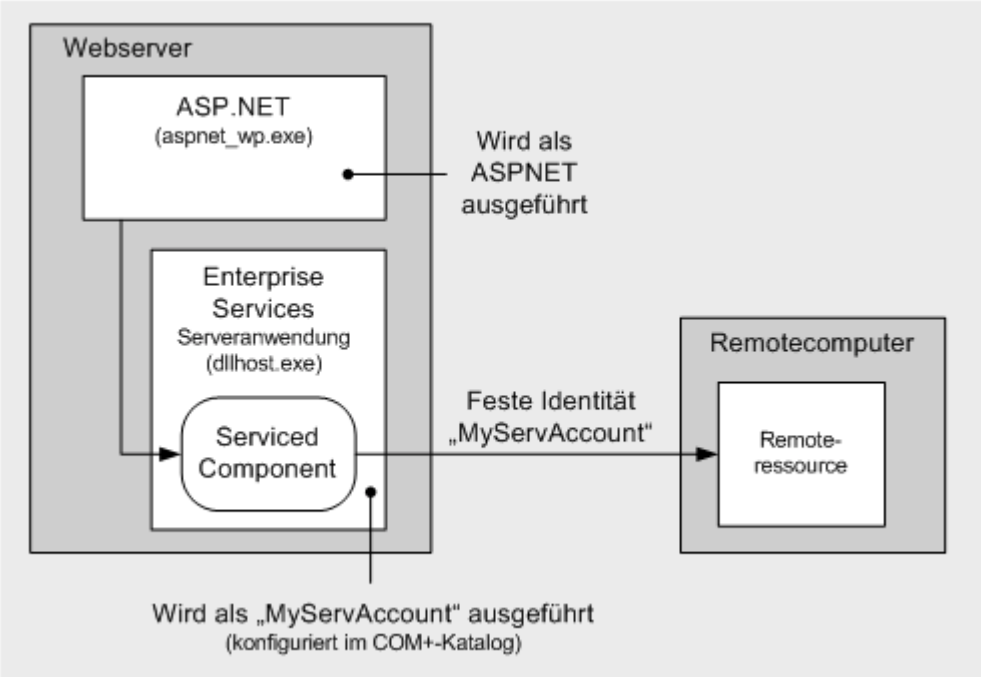

Abbildung 8.6
Verwenden einer prozessexternen Serviced Component, um eine feste Identität für den Zugriff auf Netzwerkressourcen bereitzustellen

Das Verwenden einer prozessexternen Serviced Component (in einer Enterprise Services-Serveranwendung) hat folgende Vorteile:

- Flexibilität hinsichtlich der verwendeten Identität. Keine ausschließliche Abhängigkeit von der ASP.NET-Identität.
- Vertrauenswürdiger Code oder Code mit umfangreicheren Rechten kann von der hauptsächlichen Webanwendung isoliert werden.

- Ein zusätzlicher Prozesshop erhöht die Sicherheit. Es wird für einen Angreifer wesentlich schwieriger, die Prozessgrenzen zu einem Prozess mit erhöhten Rechten zu durchbrechen.
- Wenn Sie den Identitätswechsel manuell durch Aufrufen der API **LogonUser** durchführen möchten, können Sie dies in einem Prozess erreichen, der von der Haupt-Webanwendung getrennt ist.

Hinweis: Sie müssen dem Prozesskonto für die Enterprise Services das Recht **Als Teil des Betriebssystems handeln** zuweisen, um **LogonUser** aufrufen zu können. Das Erhöhen der Rechte für einen Prozess, der von der Webanwendung getrennt ist, erweist sich weniger als Sicherheitsproblem.

Verwenden des anonymen Internetbenutzerkontos

Sie können das anonyme Internetbenutzerkonto für den Zugriff auf Netzwerkressourcen verwenden, wenn IIS für die anonyme Authentifizierung konfiguriert wurde. Dies ist der Fall, wenn eine der folgenden Optionen zutrifft:

- Ihre Anwendung unterstützt den anonymen Zugriff.
- Ihre Anwendung verwendet die Formular-, Passport- oder benutzerdefinierte Authentifizierung (wobei IIS für den anonymen Zugriff konfiguriert ist).

▶ So verwenden Sie das anonyme Konto für den Zugriff auf Remoteressourcen

1. Konfigurieren Sie IIS für die anonyme Authentifizierung. Sie können den ASP.NET-Authentifizierungsmodus in Abhängigkeit von den Authentifizierungsanforderungen der Anwendung auf **Windows**, **Formular**, **Passport** oder **Keine** festlegen.
2. Konfigurieren Sie ASP.NET für den Identitätswechsel. Verwenden Sie die folgende Einstellung in der Datei **Web.config**:

   ```
   <identity impersonate="true" />
   ```

3. Konfigurieren Sie das anonyme Konto als Domänenkonto mit minimalen Rechten.
   – oder –
   Duplizieren Sie das anonyme Konto, indem Sie denselben Benutzernamen und dasselbe Kennwort auf dem Remotecomputer verwenden. Dieser Ansatz ist erforderlich, wenn Sie Aufrufe über nicht vertrauenswürdige Domänen oder über Firewalls hinweg ausführen, bei denen die zum Unterstützen der integrierten Windows-Authentifizierung erforderlichen Anschlüsse nicht geöffnet sind.
   Auch Folgendes muss durchgeführt werden, um diesen Ansatz zu unterstützen:
   a. Verwenden Sie den Internetdienste-Manager, um das Kontrollkästchen **Kennwortkontrolle durch IIS zulassen** für das anonyme Konto zu deaktivieren.
      Wenn Sie diese Option aktivieren, erhält die mit dem angegebenen anonymen Konto erstellte Anmeldesitzung KEINE Netzwerkanmeldeinformationen (und kann daher nicht für den Zugriff auf Netzwerkressourcen verwendet werden). Wenn Sie diese Option nicht aktivieren, wird die Anmeldesitzung zu einer interaktiven Anmeldesitzung mit Netzwerkanmeldeinformationen.
   b. Legen Sie die Anmeldeinformationen des Kontos sowohl im Benutzer-Manager als auch im Internetdienste-Manager fest.

Wichtig: Wenn Sie die Identität für das anonyme Konto wechseln (z. B. IUSR_MACHINE), müssen die Ressourcen vor diesem Konto geschützt werden (mithilfe der entsprechend konfigurierten Zugriffssteuerungslisten) Ressourcen, auf die Ihre Anwendung zugreifen muss, müssen dem anonymen Konto (mindestens) den Lesezugriff gewähren. Alle anderen Ressourcen sollten dem anonymen Konto den Zugriff verweigern.

Hosten mehrerer Webanwendungen

Sie können für jedes virtuelle Stammverzeichnis Ihrer Website ein separates anonymes Internetbenutzerkonto verwenden. In einer Hostumgebung haben Sie so die Möglichkeit, Anforderungen getrennt zu autorisieren, nachzuverfolgen sowie zu überprüfen, die von separaten Webanwendungen stammen. Diese Vorgehensweise ist in Abbildung 8.7 dargestellt.

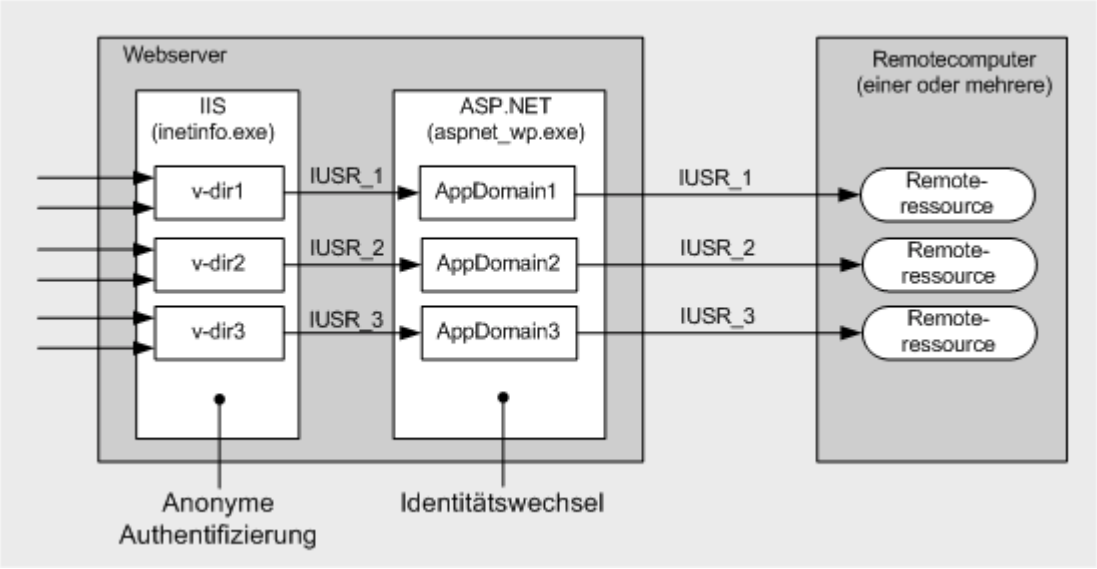

Abbildung 8.7
*Identitätswechsel für separate anonyme Internetbenutzerkonten über eine Anwendung durchführen (**v-dir**)*

▶ **So konfigurieren Sie das anonyme Internetbenutzerkonto für ein bestimmtes virtuelles Verzeichnis**

1. Starten Sie in der Programmgruppe **Verwaltung** den Internetdienste-Manager.
2. Wählen Sie das zu konfigurierende virtuelle Verzeichnis aus, und klicken Sie mit der rechten Maustaste, um die Option **Eigenschaften** auszuwählen.
3. Klicken Sie auf die Registerkarte **Verzeichnissicherheit**.
4. Klicken Sie in der Gruppe **Authentifizierung und Zugriffssteuerung** auf **Bearbeiten**.
5. Wählen Sie **Anonymer Zugriff**, und klicken Sie dann auf **Bearbeiten**.
6. Geben Sie den Benutzernamen und das Kennwort für das Konto ein, das IIS verwenden soll, wenn anonyme Benutzer eine Verbindung zur Site herstellen.
7. Stellen Sie sicher, dass die Option **Kennwortkontrolle durch IIS zulassen** NICHT aktiviert ist.

Verwenden von LogonUser und Durchführen des Identitätswechsels für eine bestimmte Windows-Identität

Sie können für eine bestimmte Identität einen Identitätswechsel durchführen, indem Sie die Attribute für Benutzername und Kennwort des Elements <**identity**> in der Datei **Web.config** konfigurieren oder die Win32® **LogonUser** API im Anwendungscode aufrufen.

Wichtig: Diese Ansätze sind nicht zu empfehlen. Sie sollten beide Ansätze auf Windows 2000-Servern vermeiden, da sie Sie zwingen, dem ASP.NET-Prozesskonto das Recht **Als Teil des Betriebssystems handeln** zu gewähren. Dadurch wird die Sicherheit Ihrer Anwendung erheblich verringert.

Windows Server 2003 wird diese Einschränkung aufheben.

Verwenden des ursprünglichen Aufrufers

Damit Sie die Identität des ursprünglichen Aufrufers für den Zugriff auf Remoteressourcen verwenden können, müssen Sie den Sicherheitskontext des Aufrufers vom Webserver an den Remotecomputer delegieren können.

Skalierbarkeitswarnung: Wenn Sie auf die Datendienstebene der Anwendung mithilfe der gewechselten Identität des ursprünglichen Aufrufers zugreifen, beeinflussen Sie die Skalierbarkeit der Anwendung erheblich, da das Pooling der Datenbankverbindung ineffektiv erfolgt. Der Sicherheitskontext für Datenbankverbindungen ist für jeden Benutzer verschieden.

Die folgenden Authentifizierungsschemas unterstützen die Delegierung:

- **Kerberos** – Weitere Informationen finden Sie unter "Vorgehensweise: Implementieren der Kerberos-Delegierung unter Windows 2000" im Abschnitt "Referenz" dieses Handbuchs.
- **Windows-Konten zugeordnete Clientzertifikate –** Die Zuordnung muss von IIS durchgeführt werden.
- **Standard** – Die Standardauthentifizierung unterstützt den Zugriff auf Remoteressourcen, da die Anmeldeinformationen des ursprünglichen Aufrufers unverschlüsselt auf dem Webserver zur Verfügung stehen. Diese können dazu verwendet werden, auf Authentifizierungsherausforderungen von Remotecomputern zu antworten.

  Die Standardauthentifizierung muss zusammen mit einer interaktiven oder Batchanmeldesitzung verwendet werden. Der Typ der Anmeldesitzung, der sich durch die Standardauthentifizierung ergibt, kann in der IIS-Metabasis konfiguriert werden. Weitere Informationen finden Sie in MSDN unter Platform SDK: Internet Information Services 5.1 (nur auf Englisch verfügbar).

  Wichtig: Die Standardauthentifizierung stellt den unsichersten Ansatz dar, der die Delegierung unterstützt. Der Grund hierfür ist, dass der Benutzername und das Kennwort unverschlüsselt vom Browser über das Netzwerk an den Server übergeben und dann auf dem Webserver zwischengespeichert werden. Sie können SSL verwenden, um die Anmeldeinformationen beim Transport zu schützen, aber Sie sollten nach Möglichkeit vermeiden, unverschlüsselte Anmeldeinformationen auf dem Webserver zwischenzuspeichern.

▶ So verwenden Sie den ursprünglichen Aufrufer für den Zugriff auf Remoteressourcen

1. Konfigurieren Sie IIS für die integrierte Windows-Authentifizierung (Kerberos), die Zertifikatsauthentifizierung (mit der IIS-Zertifikatszuordnung) oder die Standardauthentifizierung.
2. Konfigurieren Sie ASP.NET für die Windows-Authentifizierung und den Identitätswechsel.

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

3. Wenn Sie die Kerberos-Delegierung verwenden, konfigurieren Sie die Active Directory-Konten für die Delegierung.

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.
- Weitere Informationen zur IIS-Zertifikatszuordnung finden Sie unter **http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/ad/windows2000/howto/mapcerts.asp**.
- Weitere Informationen zum ASP.NET-Identitätswechsel finden Sie in MSDN unter .NET Framework Developers Guide (nur auf Englisch verfügbar).

Zugreifen auf Dateien auf einer UNC-Dateifreigabe

Wenn Ihre Anwendung über ASP.NET auf Dateien zugreifen muss, die sich auf einer UNC-Freigabe (Universal Naming Convention) befinden, ist es wichtig, dass Sie dem Ordner der Freigabe die NTFS-Berechtigungen hinzufügen. Sie müssen auch die Berechtigungen der Freigabe so festlegen, dass entweder dem ASP.NET-Prozesskonto oder der gewechselten Identität (wenn die Anwendung für den Identitätswechsel konfiguriert ist) zumindest der Lesezugriff gewährt wird.

Zugreifen auf Netzwerkressourcen, die nicht auf Windows basieren

Wenn die Anwendung auf Ressourcen zugreifen muss, die nicht auf Windows basieren, z. B. Datenbanken, die sich nicht auf Windows-Plattformen befinden, oder Großrechneranwendungen, sollten Sie die folgenden Fragen bedenken:

- Welche Gatekeeper und Vertrauensgrenzen sind dieser Ressource zugeordnet?
- Welche Anmeldeinformationen sind für die Authentifizierung erforderlich?
- Muss der Ressource die Identität des ursprünglichen Aufrufers bekannt sein, oder vertraut sie der aufrufenden Anwendung (unter Verwendung eines festen Prozesses oder einer Dienstidentität)?
- Welche Leistungseinbußen sind mit dem Einrichten von Verbindungen verbunden? Wenn damit erhebliche Einbußen verbunden sind, müssen Sie möglicherweise das Verbindungspooling implementieren, indem Sie z. B. das Objektpooling der Enterprise Services verwenden.

Wenn die Ressource in der Lage sein muss, den ursprünglichen Aufrufer zu authentifizieren (und die Windows-Authentifizierung keine Option darstellt), stehen Ihnen die folgenden Optionen zur Verfügung:

- Übergeben der Anmeldeinformationen mithilfe von Parametern (für den Methodenaufruf).
- Übergeben von Anmeldeinformationen in einer Verbindungszeichenfolge. Verwenden von SSL oder IPSec, um über ein Netzwerk gesendete unverschlüsselte Anmeldeinformationen zu schützen.

  Speichern Sie die Anmeldeinformationen sicher in Ihrer Anwendung, z. B. durch Verwenden von DPAPI. Weitere Informationen zum sicheren Speichern von Datenbank-Verbindungszeichenfolgen finden Sie unter "Sicheres Speichern von Datenbank-Verbindungszeichenfolgen" in Kapitel 12, "Datenzugriffssicherheit".
- Verwenden eines zentralen Datenspeichers für die Authentifizierung, auf den beide Plattformen zugreifen können, z. B. ein LDAP-Verzeichnis.

Sichere Kommunikation

Verwenden Sie SSL, um die Kommunikation zwischen dem Browser und dem Webserver zu sichern. SSL bietet die Vertraulichkeit und Integrität für Nachrichten. Verwenden Sie SSL und/oder IPSec, um einen sicheren Kanal vom Webserver zum Anwendungs- oder Datenbankserver bereitzustellen.

Weitere Informationen

Weitere Informationen zur sicheren Kommunikation finden Sie in Kapitel 4, "Sichere Kommunikation."

Speichern von vertraulichen Daten

Webanwendungen müssen häufig vertrauliche Daten speichern. Diese müssen vor unberechtigten Administratoren und Webbenutzern geschützt werden wie z. B.:

- **Unberechtigte Administratoren** – Unberechtigte Administratoren oder gewissenlose Benutzer sollten nicht in der Lage sein, privilegierte Informationen anzuzeigen. Der Administrator des Webservers sollte z. B. nicht das Kennwort eines Anmeldekontos für einen SQL Server lesen können, der sich auf einem Computer im Netzwerk befindet.
- **Unberechtigte Webbenutzer** – Obwohl Komponenten vorhanden sind (z. B. **FileAuthorizationModule**), die Benutzern den Zugriff auf privilegierte Dateien verwehren, wenn ein Angreifer Zugang zu einer Konfigurationsdatei erlangt, sollten die vertraulichen Daten in der Datei nicht unverschlüsselt enthalten sein.

Typische Beispiele für vertrauliche Daten sind:

- **SQL-Verbindungszeichenfolgen** – Ein häufiger Fehler ist es, den Benutzernamen und das Kennwort unverschlüsselt zu speichern. Es wird empfohlen, die Windows-Authentifizierung statt der SQL-Authentifizierung zu verwenden. Wenn Sie die Windows-Authentifizierung nicht verwenden können, lesen Sie die folgenden Abschnitte in Kapitel 12, "Datenzugriffssicherheit", die sichere Alternativen beschreiben:
  - "Sicheres Speichern von Datenbankverbindungen"
  - "Sichere Kommunikation"
- **Anmeldeinformationen für SQL-Anwendungsrollen** – SQL-Anwendungsrollen müssen über eine gespeicherte Prozedur aktiviert werden, die den Rollennamen und das zugehörige Kennwort erfordert. Weitere Informationen finden Sie unter "Autorisierung" in Kapitel 12, "Datenzugriffssicherheit".
- **Feste Identitäten in der Datei "Web.config"** – Beispiel:

  ```
  <identity impersonate=”true” userName="bob" password="inClearText"/>
  ```

  Im .NET Framework, Version 1.1, bietet ASP.NET die Möglichkeit, den Benutzernamen und das Kennwort zu verschlüsseln und sicher in einem Registrierungsschlüssel zu speichern.
- **Prozessidentität in der Datei "Machine.config"** – Beispiel:

  ```
  <process userName="cUsTuMUzerName" password=”kUsTumPazzWerD” >
  ```

  Standardmäßig kümmert sich ASP.NET um die vertraulichen Daten, wenn Sie den Benutzernamen **Machine** und das Kennwort **AutoGenerate** verwenden.

  Im .NET Framework, Version 1.1, bietet ASP.NET die Möglichkeit, den Benutzernamen und das Kennwort zu verschlüsseln und sicher in einem Registrierungsschlüssel zu speichern.

- **Schlüssel, die zum sicheren Speichern von Daten verwendet werden** – Es ist unmöglich, Schlüssel in der Software sicher zu speichern. Bestimmte Maßnahmen können dieses Risiko jedoch verringern. Ein Beispiel stellt das Erstellen eines Abschnittshandlers für benutzerdefinierte Konfigurationen dar, der Sitzungsschlüssel asymmetrisch verschlüsselt. Der Sitzungsschlüssel kann dann in einer Konfigurationsdatei gespeichert werden.
- **SQL Server-Sitzungsstatus** – Verwenden Sie die folgenden Einstellungen für die Datei **Web.config**, um SQL Server zum Verwalten des Sitzungsstatus von ASP.NET-Webanwendungen zu verwenden.

```
<sessionState … stateConnectionString="tcpip=127.0.0.1:42424"
                sqlConnectionString="data source=127.0.0.1;
                user id=UserName;password=MyPassword" />
```

  Im .NET Framework, Version 1.1, bietet ASP.NET die Möglichkeit, diese Daten zu verschlüsseln.
- **Kennwörter, die für die Formularauthentifizierung anhand einer Datenbank verwendet werden** –

  Wenn die Anwendung Anmeldeinformationen für die Authentifizierung anhand einer Datenbank überprüft, sollten Sie Kennwörter nicht in der Datenbank speichern. Verwenden Sie einen Hash des Kennworts mit einem Salt-Wert, und vergleichen Sie dann die Hashes.

  Weitere Informationen finden Sie unter "Authentifizieren von Benutzern anhand einer Datenbank" in Kapitel 12, "Datenzugriffssicherheit".

Optionen zum Speichern von vertraulichen Daten in ASP.NET

Entwicklern von .NET-Webanwendungen bieten sich eine Reihe von Ansätzen zum Speichern von vertraulichen Daten. Dabei handelt es sich um die Folgenden:

- **.NET-Kryptografieklassen** – Das .NET Framework umfasst Klassen, die zum Verschlüsseln und Entschlüsseln verwendet werden können. Diese Ansätze erfordern, dass Sie den Verschlüsselungsschlüssel sicher speichern.
- **Data Protection API (DPAPI)** – Die DPAPI besteht aus einem Paar Win32 APIs, die Daten mithilfe eines Schlüssels ver- und entschlüsseln, der vom Kennwort des Benutzers abgeleitet wird. Wenn Sie die DPAPI verwenden, müssen Sie sich nicht um die Schlüsselverwaltung kümmern. Das Betriebssystem verwaltet den Schlüssel, bei dem es sich um das Kennwort des Benutzers handelt.
- **COM+-Konstruktorzeichenfolgen** – Wenn die Anwendung Serviced Components verwendet, können Sie die vertraulichen Daten in einer Zeichenfolge zur Objektkonstruktion speichern. Die Zeichenfolge wird in unverschlüsselter Textform im COM+-Katalog gespeichert.
- **CAPICOM** – Dies ist ein Microsoft COM-Objekt, das COM-basierten Zugriff für die zugrunde liegende Crypto API bereitstellt.
- **Crypto API** – Dies sind Win32 APIs auf unterer Ebene, die die Verschlüsselung und Entschlüsselung durchführen.

Weitere Informationen

Weitere Informationen finden Sie unter dem Eintrag für Cryptography, CryptoAPI and CAPICOM im Plattform-SDK in MSDN (nur auf Englisch verfügbar).

Erwägen, vertrauliche Daten in Dateien auf separaten logischen Datenträgern zu speichern

Erwägen Sie die Installation von Webanwendungsverzeichnissen auf einem vom Betriebssystem getrennten logischen Datenträger (z. B. E: anstelle von C:). Das bedeutet, dass sich die Datei **Machine.config** (befindet sich unter **C:\WINNT\Microsoft.NET**) und möglicherweise andere Dateien, die vertrauliche Daten enthalten (z. B. UDL-Dateien), auf einem von den Webanwendungsverzeichnissen getrennten logischen Datenträger befinden.

Die Erklärung für diesen Ansatz ist der Schutz vor möglichen Fehlern bei der Dateivereinheitlichung und Verzeichnisdurchquerung. Grund:

- Fehler bei der Dateivereinheitlichung können Dateien in Ordnern von Webanwendungen offen legen.

  Hinweis: Routinen für die Dateivereinheitlichung geben die kanonische Form eines Dateipfades zurück. Hierbei handelt es sich normalerweise um den absoluten Pfadnamen, in dem alle relativen Verweise auf das aktuelle Verzeichnis vollständig aufgelöst wurden.

- Fehler bei der Verzeichnisdurchquerung können Dateien in anderen Ordnern auf demselben logischen Datenträger offen legen.

Bis jetzt wurden keine Fehler der o. a. Art veröffentlicht, die Dateien auf anderen logischen Datenträgern offen gelegt haben.

Sichern von Sitzungs- und Ansichtsstatus

Webanwendungen müssen verschiedene Statusarten verwalten, einschließlich Ansichts- und Sitzungsstatus. In diesem Abschnitt wird die sichere Statusverwaltung für ASP.NET-Webanwendungen beschrieben.

Sichern des Ansichtsstatus

Gehen Sie wie folgt vor, wenn Ihre ASP.NET-Webanwendungen den Ansichtsstatus verwenden:

- Stellen Sie die Integrität des Ansichtsstatus sicher (um sicherzustellen, dass dieser bei der Übertragung nicht verändert wird), indem Sie für **enableViewStateMac**, wie nachfolgend veranschaulicht, den Wert **true** festlegen. Dies führt dazu, dass ASP.NET einen Message Authentication Code (MAC, Nachrichtenauthentifizierungscode) für den Ansichtsstatus der Seite generiert, wenn die Seite wieder vom Client zurückübermittelt wird.

  ```
  <% @ Page enableViewStateMac=true >
  ```

- Konfigurieren Sie in der Datei **Machine.config** das **validation**-Attribut für das Element <**machineKey**>, um den für die Datenüberprüfung zu verwendenden Verschlüsselungstyp festzulegen. Bedenken Sie Folgendes:
  - Der SHA1 (Secure Hash Algorithm) erzeugt einen größeren Hash als MD5 (Message Digest 5) und wird daher als sicherer erachtet. Ein durch SHA1 oder MD5 geschützter Ansichtsstatus kann jedoch bei der Übertragung oder auf der Clientseite entschlüsselt und möglicherweise unverschlüsselt angezeigt werden.
  - Verwenden Sie 3DES (Dreifach-DES, 3 Data Encryption Standard), um Änderungen am Ansichtsstatus zu ermitteln und diesen während der Übertragung zu verschlüsseln. Wenn er sich in diesem Zustand befindet, kann er auch dann nicht unverschlüsselt angezeigt werden, wenn der Ansichtsstatus decodiert wurde.

Sichern von Cookies

Cookies, die Authentifizierungs- oder Autorisierungsdaten oder andere sensitive Daten enthalten, sollten bei der Übertragung durch SSL gesichert werden. Bei der Formularauthentifizierung kann die Methode **FormsAuthentication.Encrypt** verwendet werden, um das Authentifizierungsticket zu verschlüsseln, das zwischen Client und Server in einem Cookie übergeben wird.

Sichern des SQL-Sitzungsstatus

Der Standardhandler für den (prozessinternen) ASP.NET-Sitzungsstatus unterliegt bestimmten Einschränkungen. Er kann z. B. in einer Webfarm nicht computerübergreifend arbeiten. ASP.NET ermöglicht das Speichern des Sitzungsstatus in einer SQL Server-Datenbank, um diese Einschränkung zu umgehen.

Der SQL-Sitzungsstatus kann entweder in der Datei **Machine.config** oder **Web.config** konfiguriert werden. Die Standardeinstellung in der Datei **Machine.config** wird nachfolgend veranschaulicht.

```
<sessionState mode="InProc"
              stateConnectionString="tcpip=127.0.0.1:42424"
              stateNetworkTimeout="10"
              sqlConnectionString="data source=127.0.0.1;user id=sa;password="
              cookieless="false" timeout="20"/>
```

Standardmäßig wird das SQL-Skript **InstallSqlState.sql**, das zum Erstellen der für den SQL-Sitzungsstatus verwendeten Datenbank verwendet wird, an der folgenden Position installiert.

```
C:\WINNT\Microsoft.NET\Framework\v1.0.3705
```

Wenn Sie den SQL-Sitzungsstatus verwenden, sollten Sie zwei Probleme bedenken:

- Sie müssen die Datenbank-Verbindungszeichenfolge sichern.
- Sie müssen den Sitzungsstatus beim Durchqueren des Netzwerks sichern.

Sichern der Datenbank-Verbindungszeichenfolge

Wenn Sie die SQL-Authentifizierung zum Herstellen der Verbindung zum Server verwenden, werden die Benutzer-ID und das Kennwort, wie unten dargestellt, unverschlüsselt in der Datei **Web.config** gespeichert.

```
<sessionState
        cookieless="false"
        timeout="20"
        mode="InProc"
        stateConnectionString="tcpip=127.0.0.1:42424"
        sqlConnectionString=
          "data source=127.0.0.1;user id=UserName;password=ClearTxtPassword" />
```

HttpForbiddenHandler verhindert standardmäßig, dass Konfigurationsdateien gedownloadet werden. Dennoch kann jeder Benutzer, der direkten Zugriff auf die Ordner besitzt, in denen die Konfigurationsdateien gespeichert werden, weiterhin den Benutzernamen und das Kennwort anzeigen. Eine bessere Lösung stellt das Verwenden der Windows-Authentifizierung für SQL Server dar.

▶ **Sie können die ASP.NET-Prozessidentität (normalerweise ASPNET) verwenden, um die Windows-Authentifizierung zu verwenden**

1. Erstellen Sie auf dem Datenbankserver ein Duplikat des Kontos (mit demselben Benutzernamen und demselben Kennwort).
2. Erstellen Sie für das Konto einen SQL-Benutzernamen.
3. Erstellen Sie in der Datenbank **ASPState** einen Datenbankbenutzer, und ordnen Sie den SQL-Benutzernamen dem neuen Benutzer zu.
   Die Datenbank **ASPState** wird vom Skript **InstallSQLState.sql** erstellt.
5. Erstellen Sie eine benutzerdefinierte Datenbankrolle, und fügen Sie den Datenbankbenutzer zur Rolle hinzu.
6. Konfigurieren Sie die Berechtigungen für die Datenbankrolle in der Datenbank.

Sie können dann die Verbindungszeichenfolge ändern, damit diese eine vertrauenswürdige Verbindung verwendet, wie nachfolgend gezeigt:

```
sqlConnectionString="server=127.0.0.1;
                     database=StateDatabase;
                     Integrated Security=SSPI;"
```

Sichern des Sitzungsstatus im Netzwerk

Sie müssen den Sitzungsstatus möglicherweise sichern, während dieser das Netzwerk auf dem Weg zur SQL Server-Datenbank durchquert. Dies hängt davon ab, wie sicher das Netzwerk ist, das den Webserver und die Datenserver verwaltet. Wenn die Datenbank in einer vertrauenswürdigen Umgebung physisch gesichert ist, können Sie diese zusätzliche Sicherheitsmaßnahme möglicherweise vernachlässigen.

Sie können IPSec verwenden, um den gesamten IP-Verkehr zwischen den Webservern und SQL Server zu schützen, oder Sie verwenden SSL, um die Verknüpfung zu SQL Server zu sichern. Bei diesem Ansatz bietet sich Ihnen die Möglichkeit, nur die für den Sitzungsstatus verwendete Verbindung und nicht sämtlichen Verkehr zwischen den Computern zu verschlüsseln.

Weitere Informationen

- Weitere Informationen zum Einrichten des SQL-Sitzungsstatus finden Sie in der Microsoft Knowledge Base im Artikel Q317604, "HOW TO: Configure SQL Server to Store ASP.NET Session State" (US).
- Weitere Informationen zum Verwenden von SSL für SQL Server finden Sie unter "Vorgehensweise: Verwenden von SSL zum Sichern der Kommunikation mit SQL Server 2000" im Abschnitt "Referenz" dieses Handbuchs.
- Weitere Informationen zum Verwenden von IPSec finden Sie unter "Vorgehensweise: Verwenden von IPSec zum Sichern der Kommunikation zwischen zwei Servern" im Abschnitt "Referenz" dieses Handbuchs.

Überlegungen zur Webfarm

In einem Webfarmszenario gibt es keine Garantie, dass nachfolgende Anforderungen desselben Clients auch von demselben Webserver bearbeitet werden. Dies hat Auswirkungen auf die Statusverwaltung sowie auf sämtliche Verschlüsselungen, die von Attributen abhängen, die vom Element <**machineKey**> in der Datei **Machine.config** verwaltet werden.

Sitzungsstatus

Die prozessinterne Standardbehandlung für den ASP.NET-Sitzungsstatus (die frühere ASP-Funktionen widerspiegelt) führt zu Serveraffinität und kann nicht in einem Webfarmszenario verwendet werden. Bei der Webfarmbereitstellung muss der Sitzungsstatus, wie zuvor beschrieben, prozessextern entweder im ASP.NET-Statusdienst oder in einer SQL Server-Datenbank gespeichert werden.

Hinweis: Sie können sich beim Verwalten globaler Indikatoren oder eindeutiger Werte in Webfarm- (zum Ausführen auf mehreren Servern konfigurierte Webanwendung) oder Webgartenszenarien (zum Ausführen auf mehreren Prozessoren konfigurierte Webanwendung) nicht auf den Anwendungsstatus verlassen, da dieser nicht prozess- oder computerübergreifend freigegeben ist.

DPAPI

DPAPI kann entweder mit dem Computer- oder Benutzerspeicher (der ein geladenes Benutzerprofil erfordert) arbeiten. Wenn Sie DPAPI mit dem Computerspeicher verwenden, ist die verschlüsselte Zeichenfolge für einen angegebenen Computer bestimmt, und daher müssen Sie die verschlüsselten Daten auf jedem Computer generieren. Kopieren Sie die verschlüsselten Daten in einer Webfarm oder einem Cluster nicht von einem Computer auf einen anderen.

Wenn Sie DPAPI mit dem Benutzerspeicher verwenden, können Sie die Daten auf einem beliebigen Computer mit einem servergespeicherten Benutzerprofil entschlüsseln.

Weitere Informationen

Weitere Informationen zu DPAPI finden Sie in Kapitel 12, "Datenzugriffssicherheit".

Verwenden der Formularauthentifizierung in einer Webfarm

Wenn Sie die Formularauthentifizierung verwenden, ist es von größter Bedeutung, dass alle Server in der Webfarm einen gemeinsamen Computerschlüssel gemeinsam nutzen, der für die Verschlüsselung, Entschlüsselung und zur Überprüfung des Authentifizierungstickets verwendet wird.

Der Computerschlüssel wird in der Datei **Machine.config** vom Element <**machineKey**> verwaltet. Nachfolgend wird die Standardeinstellung angezeigt.

```
<machineKey validationKey="AutoGenerate"
            decryptionKey="AutoGenerate"
            validation="SHA1"/>
```

Diese Einstellung führt dazu, dass jeder Computer einen anderen Überprüfungs- und Entschlüsselungsschlüssel generiert. Sie müssen das Element <**machineKey**> ändern und gemeinsame Schlüsselwerte auf allen Servern in der Webfarm positionieren.

Das Element <machineKey>

Das in der Datei **Machine.config** enthaltene Element <**machineKey**> wird dazu verwendet, die Schlüssel zu konfigurieren, die zum Verschlüsseln und Entschlüsseln von Cookiedaten und des Ansichtsstatus der Formularauthentifizierung verwendet werden.

Wenn die Methode **FormsAuthentication.Encrypt** oder **FormsAuthentication.Decrypt** aufgerufen und der Ansichtsstatus erstellt oder abgerufen wird, werden die Werte im Element <**machineKey**> herangezogen.

```
<machineKey validationKey="autogenerate|value"
            decryptionKey="autogenerate|value"
            validation="SHA1|MD5|3DES" />
```

validationKey-Attribut

Der Wert für das **validationKey**-Attribut wird verwendet, um Nachrichtenauthentifizierungscodes für Ansichtsstatus- und Formularauthentifizierungstickets zu erstellen und zu überprüfen. Das Überprüfungsattribut kennzeichnet den für die MAC-Generierung zu verwendenden Algorithmus. Beachten Sie Folgendes:

- Bei der Formularauthentifizierung arbeitet dieser Schlüssel in Verbindung mit dem <**forms**> **protection**-Attribut. Wenn das **protection**-Attribut den Wert **Validation** erhält und dann die Methode **FormsAuthentication.Encrypt** aufgerufen wird, erfolgt die Berechnung eines Nachrichtenauthentifizierungscodes, der an das Cookie angehängt wird, anhand des Ticketwerts und des Elements **validationKey**. Wenn die Methode **FormsAuthentication.Decrypt** aufgerufen wird, wird der Nachrichtenauthentifizierungscode berechnet und mit dem Nachrichtenauthentifizierungscode verglichen, der an das Ticket angehängt wird.
- Beim Ansichtsstatus wird der Nachrichtenauthentifizierungscode aus dem Wert vom Ansichtsstatus eines Steuerelements und aus **validationKey** berechnet und an den Ansichtsstatus angehängt. Wenn der Ansichtsstatus wieder vom Client zurückübermittelt wird, wird der Nachrichtenauthentifizierungscode erneut berechnet und mit dem Nachrichtenauthentifizierungscode verglichen, der an den Ansichtsstatus angehängt ist.

decryptionKey-Attribut

Der Wert für das d**ecryptionKey**-Attribut wird dazu verwendet, um Formularauthentifizierungstickets und den Ansichtsstatus zu ver- und entschlüsseln. Dazu wird der Algorithmus DES oder Dreifach-DES (3DES) verwendet. Der genaue Algorithmus hängt davon ab, ob das Windows 2000 High Encryption Pack auf dem Server installiert ist. Wenn es installiert ist, wird 3DES verwendet, andernfalls DES. Beachten Sie Folgendes:

- Bei der Formularauthentifizierung arbeitet der Schlüssel in Verbindung mit dem <**forms**> **protection**-Attribut. Wenn für das **protection**-Attribut der Wert **Encryption** festgelegt ist und die Methode **FormsAuthentication.Encrypt** oder **Decrypt** aufgerufen wird, erfolgt die Verschlüsselung oder Entschlüsselung des Ticketwerts mit dem angegebenen Wert für **decryptionKey**.
- Beim Ansichtsstatus wird der Wert vom Ansichtsstatus eines Steuerelements mit dem Wert von **decryptionKey** verschlüsselt und an den Client gesendet. Er wird wieder entschlüsselt, wenn der Client die Daten an den Server zurückübermittelt.

Validation-Attribut

Dieses Attribut legt fest, welcher Algorithmus zum Überprüfen, Verschlüsseln oder Entschlüsseln verwendet wird. Es kann die Werte SHA1, MD5 oder 3DES annehmen. Diese Werte werden nachstehend beschrieben:

- **SHA1** – Der Algorithmus HMACSHA1 wird verwendet, wenn die Einstellung SHA1 ausgewählt wurde. Er erzeugt einen 160 Bit-Hash (20 Byte) oder Digest der Eingabe. HMACSHA1 ist ein verschlüsselter Hashalgorithmus. Der für diesen Algorithmus als Eingabe verwendete Schlüssel wird vom **validationKey**-Attribut festgelegt.
  SHA1 ist ein gebräuchlicher Algorithmus, da er im Vergleich zu anderen Algorithmen einen größeren Digest verwendet.
- **MD5** – Erzeugt mit dem MD5-Algorithmus einen Hash mit einer Größe von 20 Byte.
- **3DES** – Verschlüsselt Daten mit dem Dreifach-DES-Algorithmus (3DES).

Hinweis: Wenn das Überprüfungsattribut den Wert 3DES erhält, wird dieser Algorithmus nicht bei der Formularauthentifizierung verwendet. Stattdessen wird SHA1 verwendet.

Weitere Informationen

- Weitere Informationen zum Erstellen von Schlüsseln, die für die Datei **Machine.config** geeignet sind, finden Sie in der Microsoft Knowledge Base im Artikel Q312906, "HOW TO: Create Keys w/ C# .NET for Use in Forms Authentication" (US).
- Weitere Informationen zum Windows 2000 High Encryption Pack finden Sie unter http://www.microsoft.com/windows2000/downloads/recommended/encryption/.

Zusammenfassung

In diesem Kapitel wurden eine Vielzahl von Verfahren und Ansätzen zum Sichern von ASP.NET-Webanwendungen beschrieben. Viele der in diesem Kapitel bereitgestellten Anleitungen und Empfehlungen gelten auch für die Entwicklung von ASP.NET-Webdiensten und .NET Remoting-Objekten, die von ASP.NET verwaltet werden. Zusammengefasst:

- Wenn Ihre Anwendung die Formularauthentifizierung verwendet und die Leistung bei der Authentifizierung des Benutzers von Bedeutung ist, rufen Sie eine Liste mit Rollen ab, und speichern Sie diese im Authentifizierungsticket.
- Wenn Sie die Formularauthentifizierung verwenden, erstellen Sie immer einen Principal, und speichern Sie diesen im Kontext jeder Anforderung.
- Wenn in einem Authentifizierungscookie zu viele Rollen gespeichert werden müssen, dann verwenden Sie den globalen Anwendungscache, um die Rollen zu speichern.
- Verwenden Sie zum Ausführen von ASP.NET kein benutzerdefiniertes Konto mit minimalen Rechten. Ändern Sie stattdessen das ASPNET-Kontokennwort, und erstellen Sie auf jedem Windows-Remoteserver, auf den die Anwendung zugreifen muss, ein Duplikat des Kontos.
- Wenn Sie ein benutzerdefiniertes Konto erstellen müssen, um ASP.NET auszuführen, verwenden Sie das Prinzip der minimalen Rechte. Beispiel:
  - Verwenden Sie ein Domänenkonto mit minimalen Rechten, wenn der Verwaltung das Hauptinteresse gilt.
  - Wenn Sie ein lokales Konto verwenden, müssen Sie auf allen Remotecomputern, auf die die Webanwendung zugreifen muss, ein Duplikat des Kontos erstellen. Sie müssen lokale Konten verwenden, wenn die Anwendung auf Ressourcen in nicht vertrauenswürdigen Domänen zugreifen muss oder wo ein Firewall die Windows-Authentifizierung verhindert.
  - Führen Sie ASP.NET nicht unter Verwendung des lokalen Kontos SYSTEM aus.
  - Erteilen Sie dem ASPNET-Konto nicht das Recht **Als Teil des Betriebssystems handeln**.
- Verwenden Sie unter den folgenden Bedingungen SSL:
  - Zwischen dem Browser und dem Webserver werden sicherheitsrelevante Daten ausgetauscht.
  - Die Standardauthentifizierung wird verwendet (zum Schützen von Anmeldeinformationen).
  - Wenn die Formularauthentifizierung für die Authentifizierung verwendet wird (im Gegensatz zur Personalisierung).
- Vermeiden Sie es, vertrauliche Daten unverschlüsselt zu speichern.