Erstellen sicherer ASP.NET-Anwendungen

Authentifizierung, Autorisierung und sichere Kommunikation

Kapitel 3 – Authentifizierung und Autorisierung

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

Dieses Kapitel enthält hilfreiche Informationen für die Entwicklung einer geeigneten Autorisierungsstrategie für das jeweilige Anwendungsszenario. Sie erhalten Hilfestellungen bei der Auswahl der geeignetesten Authentifizierungs- und Autorisierungstechnik und der Anwendung dieser Technik an den richtigen Stellen in der Anwendung.

Inhalt

Der Entwurf einer Authentifizierungs- und Autorisierungsstrategie für verteilte Webanwendungen ist eine anspruchsvolle Aufgabe. Ein einwandfreier Authentifizierungs- und Autorisierungsentwurf in den frühen Phasen der Anwendungsentwicklung hilft jedoch, viele häufige Sicherheitsrisiken zu mildern.

Dieses Kapitel unterstützt Sie beim Entwurf einer geeigneten Autorisierungsstrategie für die Anwendung und bei der Beantwortung der folgenden Schlüsselfragen:

- Wo soll die Autorisierung stattfinden, und welche Mechanismen sollen verwendet werden?
- Welche Authentifizierungsmechanismen sollen verwendet werden?
- Soll der Verzeichnisdienst Active Directory® für die Authentifizierung verwendet werden, oder sollen die Anmeldeinformationen anhand eines benutzerdefinierten Datenspeichers überprüft werden?
- Welche Auswirkungen und Entwurfskriterien müssen bei heterogenen und homogenen Plattformen berücksichtigt werden?
- Wie sollen Benutzer in der Anwendung dargestellt werden, die kein Microsoft® Windows®-Betriebssystem verwenden?
- Wie soll die Benutzeridentität durch die Ebenen der Anwendung übermittelt werden? Wann sollen Identitätswechsel/Delegierungen auf Betriebssystemebene verwendet werden?

Bei der Betrachtung der Autorisierung muss auch die Authentifizierung berücksichtigt werden. Die beiden Prozesse sind aus zwei Gründen voneinander abhängig:

- Zunächst setzt jede sinnvolle Autorisierungsrichtlinie authentifizierte Benutzer voraus.
- Zweitens bestimmt die Art der Authentifizierung der Benutzer (und insbesondere die Art der Darstellung der authentifizierten Benutzeridentität in der Anwendung) die Ihnen zur Verfügung stehenden Gatekeeper.

  Einige Gatekeeper wie die ASP.NET-Dateiautorisierung, Enterprise Services (COM+)-Rollen und Windows-ACLs erfordern eine authentifizierte Windows-Identität (in der Form eines **WindowsIdentity**-Objekts, das ein Windows-Zugriffstoken kapselt, das den Sicherheitskontext des Aufrufers definiert). Andere Gatekeeper, wie z. B. die URL-Autorisierung von ASP.NET und .NET-Rollen, setzen lediglich eine authentifizierte Identität voraus, die nicht notwendigerweise durch ein Windows-Zugriffstoken dargestellt wird.

Entwerfen einer Authentifizierungs- und Autorisierungsstrategie

Die folgenden Schritte beschreiben eine Vorgehensweise, die Sie beim Entwickeln einer Authentifizierungs- und Autorisierungsstrategie für die Anwendung unterstützt:

1. Identifizieren der Ressourcen
2. Auswählen einer Autorisierungsstrategie
3. Auswählen der für den Ressourcenzugriff verwendeten Identitäten
4. Berücksichtigen der Identitätsübermittlung
5. Auswählen eines Authentifizierungsverfahrens
6. Festlegen der Identitätsübermittlung

Identifizieren der Ressourcen

Identifizieren Sie die Ressourcen, die die Anwendung Clients gegenüber offenlegen muss. Dabei handelt es sich normalerweise um die folgenden Ressourcen:

- Webserverressourcen, z. B. Webseiten, Webdienste, statische Ressourcen (HTML-Seiten und Bilder).
- Datenbankressource, z. B. benutzerspezifische Daten oder anwendungsweite Daten.
- Netzwerkressourcen, z. B. Remotedateisystemressourcen und Daten aus Verzeichnisspeichern wie Active Directory.

Sie müssen außerdem die Systemressourcen identifizieren, auf die die Anwendung zugreifen können muss (im Gegensatz zu Ressourcen, die gegenüber Clients offengelegt werden). Beispiele für Systemressourcen sind Registrierung, Ereignisprotokolle und Konfigurationsdateien.

Auswählen einer Autorisierungsstrategie

Es gibt die beiden folgenden grundlegenden Autorisierungsstrategien:

- **Rollenbasiert -** Der Zugriff auf Operationen (normalerweise Methoden) wird anhand der Rollenmitgliedschaft des Aufrufers gesichert. Mit Rollen wird die Benutzerbasis der Anwendung in Benutzergruppen mit gleichen Sicherheitsrechten in der Anwendung aufgeteilt, z. B. Geschäftsleitung, Abteilungsleiter und Mitarbeiter. Die Benutzer werden Rollen zugewiesen. Wenn der Benutzer autorisiert ist, die angeforderte Operation auszuführen, verwendet die Anwendung feste Identitäten für den Zugriff auf die Ressourcen. Diesen Identitäten wird von der entsprechenden Ressourcenverwaltung (z. B. Datenbanken, Dateisystem usw.) vertraut.
- **Ressourcenbasiert** - Die einzelnen Ressourcen werden mithilfe von Windows-ACLs gesichert. Die Anwendung wechselt die Identität des Aufrufers vor dem Zugriff auf die Ressourcen, sodass das Betriebssystem die normalen Zugriffsprüfungen durchführen kann. Jeder Ressourcenzugriff erfolgt im Sicherheitskontext des ursprünglichen Aufrufers. Identitätswechsel wirken sich erheblich auf die Skalierbarkeit der Anwendung aus, da in der mittleren Ebene der Anwendung kein effektives Verbindungspooling verwendet werden kann.

Bei der Mehrheit der .NET-Webanwendungen, bei denen die Skalierbarkeit von Bedeutung ist, stellt ein rollenbasiertes Verfahren für die Autorisierung die optimale Möglichkeit dar. Bei bestimmten kleineren Intranetanwendungen, die einzelnen Benutzern Inhalte aus Ressourcen (z. B. Dateien) bereitstellen, und die mit Windows-ACLs bezüglich einzelner Benutzer gesichert werden können, ist eine ressourcenbasierte Vorgehensweise u. U. am geeignetsten.

Für eine rollenbasierte Autorisierung wird das folgende allgemeingültige Muster empfohlen:

- Authentifizieren der Benutzer in der Front-End-Webanwendung.
- Zuordnen der Benutzer zu Rollen.
- Autorisieren des Zugriffs auf Operationen (nicht direkt auf Ressourcen) anhand der Rollenmitgliedschaft.
- Zugriff auf die erforderlichen Back-End-Ressourcen (die für die Unterstützung der angeforderten und autorisierten Operationen erforderlich sind) mithilfe von festen Dienstidentitäten. Die Verwaltungen der Back-End-Ressourcen (z. B. Datenbanken) *vertrauen* der Anwendung hinsichtlich der Autorisierung der Aufrufer und gewähren den vertrauten Dienstidentitäten die entsprechenden Berechtigungen.

  Ein Datenbankadministrator kann beispielsweise ausschließlich einer bestimmten Personalanwendung (aber nicht einzelnen Benutzern) Zugriffsrechte gewähren.

Weitere Informationen

- Weitere Informationen zu den beiden gegensätzlichen Autorisierungsverfahren finden Sie weiter unten in diesem Kapitel unter "Autorisierungsverfahren".
- Weitere Informationen zur rollenbasierten Autorisierung und den verschiedenen Typen der verwendbaren Rollen finden Sie weiter unten in diesem Kapitel unter "Rollenbasierte Autorisierung".

Auswählen der für den Ressourcenzugriff verwendeten Identitäten

Beantworten Sie die Frage: „Wer greift auf die Ressourcen zu?“.

Wählen Sie die Identität bzw. die Identitäten aus, die durch die Ebenen der Anwendung für den Zugriff auf die Ressourcen verwendet werden sollen. Dies beinhaltet Ressourcen, auf die von webbasierten Anwendungen zugegriffen wird, und optional Webdienste, Enterprise Services und .NET Remoting-Komponenten. In allen Fällen kommen für den Ressourcenzugriff die folgenden Identitäten in Frage:

- **Die Identität des ursprünglichen Aufrufers** - Dies setzt ein Modell mit Identitätswechsel/Delegierung voraus, in dem die Identität des ursprünglichen Aufrufers ermittelt werden und anschließend durch die einzelnen Schichten des Systems laufen kann. Der Delegierungsfaktor bildet ein wesentliches Kriterium, anhand dessen der Authentifizierungsmechanismus bestimmt wird.
- **Prozessidentität** - Hierbei handelt es sich um den Standardfall (ohne spezifischen Identitätswechsel). Zugriffe auf lokale Ressourcen und nachgeschaltete Aufrufe erfolgen mit der aktuellen Prozessidentität. Die Machbarkeit dieses Verfahrens hängt von den überschrittenen Grenzen ab, da die Prozessidentität vom Zielsystem erkannt werden muss.

  Dies impliziert, dass die Aufrufe auf eine der folgenden Arten durchgeführt werden:

  - Innerhalb derselben Windows-Sicherheitsdomäne
  - Übergreifend über Windows-Sicherheitsdomänen (unter Verwendung von Vertrauensstellungen und Domänenkonten bzw. duplizierter Benutzernamen und Kennwörter, sofern keine Vertrauensbeziehung besteht)

- **Dienstkonto** - Bei diesem Verfahren wird ein (festes) Dienstkonto verwendet. Beispiel:

  - Beim Datenbankzugriff kann es sich um einen festen SQL-Benutzernamen mit Kennwort handeln, die von der Komponente übergeben werden, die eine Verbindung zu der Datenbank herstellen.
  - Verwenden Sie eine Enterprise Services-Serveranwendung, wenn eine feste Windows-Identität erforderlich ist.

- **Benutzerdefinierte Identität** - Wenn keine Windows-Konten vorhanden sind, können Sie eigene Identitäten erstellen (mit den **IPrincipal**- und **IIdentity**-Implementierungen), die Details im Hinblick auf den eigenen jeweiligen Sicherheitskontext enthalten können. Dafür können beispielsweise Rollenlisten, eindeutige Bezeichner oder beliebige andere benutzerdefinierte Informationen verwendet werden.

  Durch Implementieren der benutzerdefinierten Identität mit den Typen **IPrincipal** und **IIdentity** und deren Einbringen in den aktuellen Webkontext (mit **HttpContext.User**) profitieren Sie unmittelbar von den integrierten Gatekeepern wie .NET-Rollen und **PrincipalPermission**-Forderungen.

Berücksichtigen der Identitätsübermittlung

Um eine Autorisierung der einzelnen Benutzer, Überwachung sowie einen Datenabruf der einzelnen Benutzer zu unterstützen, muss die Identität des ursprünglichen Aufrufers durch die Anwendungsebenen und über mehrere Computergrenzen hinweg übermittelt werden. Wenn eine Back-End-Ressourcenverwaltung beispielsweise eine Autorisierung einzelner Benutzer durchführen muss, muss die Identität des Aufrufers an die betreffende Ressourcenverwaltung übergeben werden.

Abhängig von den Anforderungen des Systems im Hinblick auf die Autorisierung durch die Ressourcenverwaltung und die Überwachung müssen Sie die Identitäten identifizieren, die durch die Anwendung übergeben werden müssen.

Auswählen eines Authentifizierungsverfahrens

Zweie wesentliche Faktoren beeinflussen die Auswahl des Authentifizierungsverfahrens: zuerst das Wesen der Benutzerbasis der Anwendung (die verwendeten Browsertypen und das Vorhandensein von Windows-Konten) und zweitens die Anforderungen der Anwendung im Hinblick auf Identitätswechsel/Delegierung und Überwachung.

Weitere Informationen

Ausführlichere Überlegungen, die Ihnen bei der Auswahl eines Authentifizierungsmechanismus für die Anwendung helfen, finden Sie weiter unten in diesem Kapitel unter "Auswählen eines Authentifizierungsmechanismus".

Festlegen der Identitätsübermittlung

Die Identität kann auf Anwendungsebene übermittelt werden (um den Sicherheitskontext bereitzustellen), Identität und Sicherheitskontext können aber auch auf Betriebssystemebene übermittelt werden.

Wenn die Identität auf Anwendungsebene übermittelt werden soll, verwenden Sie Parameter von Methoden und gespeicherten Prozeduren. Die Identitätsübermittlung auf Anwendungsebene unterstützt:

- Datenabruf einzelner Benutzer mithilfe von vertrauten Abfrageparametern

```
SELECT x,y FROM SomeTable WHERE username="bob"
```

- Benutzerdefinierte Überwachung in jeder Anwendungsebene

Die Identitätsübermittlung auf Betriebssystemebene unterstützt:

- Überwachung auf Plattformebene (z. B. Windows-Überwachung und SQL Server-Überwachung)
- Autorisierung einzelner Benutzer basierend auf Windows-Identitäten

Für die Identitätsübermittlung auf Betriebssystemebene können Sie das Modell mit Identitätswechsel/Delegierung verwenden. In bestimmten Fällen können Sie die Kerberos-Delegierung verwenden, während Sie in anderen Fällen (in denen Kerberos u. U. von der Umgebung nicht unterstützt wird) evtl. andere Verfahren verwenden müssen, z. B. die Standardauthentifizierung. Die Anmeldeinformationen des Benutzers stehen bei der Standardauthentifizierung der Serveranwendung zur Verfügung und können für den Zugriff auf nachgeschaltete Netzwerkressourcen verwendet werden.

Weitere Informationen

Weitere Informationen zur Identitätsübermittlung und zum Ermitteln eines Identitätswechseltokens mit Netzwerkanmeldeinformationen (d. h. eine Delegierung wird unterstützt) finden Sie weiter unten in diesem Kapitel unter "Identitätsübermittlung".

Autorisierungsverfahren

Es gibt zwei grundlegende Autorisierungsverfahren:

- **Rollenbasiert** - Die Benutzer werden in anwendungsdefinierte, logische Rollen aufgeteilt. Die Mitglieder einer bestimmten Rolle verfügen innerhalb der Anwendung über die gleichen Rechte. Der Zugriff auf Operationen (normalerweise durch Methodenaufrufe ausgedrückt) wird anhand der Rollenmitgliedschaft des Aufrufers autorisiert.

  Der Zugriff auf Ressourcen erfolgt mit festen Identitäten (z. B. die Prozessidentität einer Webanwendung oder eines Webdienstes). Die Ressourceverwaltungen vertrauen darauf, dass die Anwendung die Benutzer ordnungsgemäß autorisiert, und autorisieren die *vertrauenswürdige* Identität.
- **Ressourcenbasiert** - Die einzelnen Ressourcen werden mithilfe von Windows-ACLs gesichert. Die ACL bestimmt die Benutzer, die auf die Ressource zugreifen dürfen, und ebenfalls die Art der Operationen, die die einzelnen Benutzer ausführen dürfen (Lesen, Schreiben, Löschen usw.).

  Der Zugriff auf die Ressourcen erfolgt mit der Identität des ursprünglichen Aufrufers (unter Verwendung von Identitätswechseln).

Rollenbasiert

Bei einem rollenbasierten (oder operationsbasierten) Sicherheitsverfahren wird der Zugriff auf Operationen (nicht auf Back-End-Ressourcen) anhand der Rollenmitgliedschaft des Aufrufers autorisiert. Die Rollen (die zur Entwurfszeit der Anwendung analysiert und definiert werden) werden als logische Container verwendet, in denen Benutzer mit denselben Sicherheitsrechten (oder -merkmalen) in der Anwendung gruppiert werden. Die Benutzer werden den Rollen in der Anwendung zugeordnet. Anhand der Rollenmitgliedschaft wird der Zugriff auf bestimmte Operationen (Methoden) gesteuert, die von der Anwendung offengelegt werden.

An welcher Stelle in der Anwendung die Rollenzuordnung erfolgt, bildet ein wesentliches Entwurfskriterium, z. B.:

- Die Rollenzuordnung kann einerseits in der Verwaltung einer Back-End-Ressource, z. B. einer Datenbank, durchgeführt werden. Dabei muss der Sicherheitskontext des ursprünglichen Aufrufers die Anwendungsebenen bis zu Back-End-Datenbank durchlaufen.
- Die Rollenzuordnung kann andererseits in der Front-End-Webanwendung durchgeführt werden. Bei dieser Vorgehensweise erfolgt der Zugriff auf die nachgeschalteten Ressourcenverwaltungen mit festen Identitäten, die von der Ressourcenverwaltung autorisiert und denen von der Ressourcenverwaltung vertraut wird.
- Eine dritte Möglichkeit besteht darin, die Rollenzuordnung zwischen der Front-End- und der Back-End-Ebene durchzuführen, beispielsweise in der mittleren Ebene einer Enterprise Services-Anwendung.

In mehrstufigen Webanwendungen bietet die Verwendung vertrauter Identitäten für den Zugriff auf die Verwaltung von Back-End-Ressourcen bessere Skalierungsmöglichkeiten für die Anwendung (aufgrund des Verbindungspoolings). Die Verwendung vertrauter Identitäten reduziert außerdem die Notwendigkeit, den Sicherheitskontext des ursprünglichen Aufrufers auf Betriebssystemebene zu übermitteln, was u. U. schwer zu erreichen ist (wenn nicht in bestimmten Szenarien sogar unmöglich).

Ressourcenbasiert

Bei ressourcenbasierten Autorisierungsverfahren werden Windows-ACLs und die zugrunde liegende Zugriffssteuerungsmechanik des Betriebssystems verwendet. Die Anwendung wechselt die Identität des Aufrufers und überlässt es dem Betriebssystem, in Verbindung mit den speziellen Ressourcenverwaltungen (Dateisystem, Datenbanken usw.) die Zugriffsprüfungen durchzuführen.

Dieses Verfahren eignet sich optimal für Anwendungen, die Zugriff auf Ressourcen ermöglichen, die individuell mit Windows-ACLs gesichert werden können, z. B. Dateien. Beispiel wäre eine FTP-Anwendung oder eine datengesteuerte Webanwendung. Das Verfahren wird dann unbrauchbar, wenn die angeforderte Ressource aus Daten besteht, die von mehreren verschiedenen Quellen abgerufen und konsolidiert werden müssen, z. B. mehrere Datenbanken, Datenbanktabellen, externe Anwendungen oder Webdienste.

Beim ressourcenbasierten Verfahren muss ebenfalls der Sicherheitskontext des ursprünglichen Aufrufers durch die Anwendung an die Verwaltungen der Back-End-Ressourcen übermittelt werden. Dies kann eine komplexe Konfiguration erforderlich machen und reduziert die Skalierungsmöglichkeit einer mehrstufigen Anwendung für eine Vielzahl von Benutzern erheblich, da in der mittleren Ebene der Anwendung ein effizienter Poolingeinsatz (z. B. Datenbankverbindungspooling) verhindert wird.

Modelle für den Ressourcenzugriff

Die beiden gegensätzlichen Autorisierungsverfahren können anhand der beiden von .NET-Webanwendungen (und allgemein von verteilten mehrstufigen Anwendungen) am häufigsten verwendeten Sicherheitsmodelle für den Ressourcenzugriff verdeutlicht werden. Diese sind:

- Das Modell mit vertrauenswürdigen Subsystemen
- Das Modell mit Identitätswechsel/Delegierung

Jedes Model weist im Hinblick auf Sicherheit und Skalierbarkeit Vor- und Nachteile auf. Diese Modelle werden in den nächsten Abschnitten beschrieben.

Das Modell mit vertrauenswürdigen Subsystemen

Bei diesem Modell verwendet der Dienst auf der mittleren Ebene für den Zugriff auf nachgeschaltete Dienste und Ressourcen eine feste Identität. Der Sicherheitskontext des ursprünglichen Aufrufers wird nicht auf Betriebssystemebene durch den Dienst übertragen, obwohl die Anwendung die Identität des ursprünglichen Aufrufers bei Bedarf auf Anwendungsebene übermitteln kann. Das kann erforderlich sein, um die Anforderungen der Back-End-Überwachung oder Datenzugriff und Autorisierung einzelner Benutzer zu unterstützen.

Der Name des Modells entstammt der Tatsache, dass der nachgeschaltete Dienst (möglicherweise eine Datenbank) dem vorgeschalteten Dienst vertraut, dass die Aufrufer autorisiert werden. Dieses Modell ist in Abbildung 3.1 dargestellt. Achten Sie besonders auf die Vertrauensgrenze. In diesem Beispiel *vertraut* die Datenbank der mittleren Ebene, dass die Aufrufer autorisiert werden und nur autorisierte Aufrufer mit der vertrauenswürdigen Identität auf die Datenbank zugreifen können.

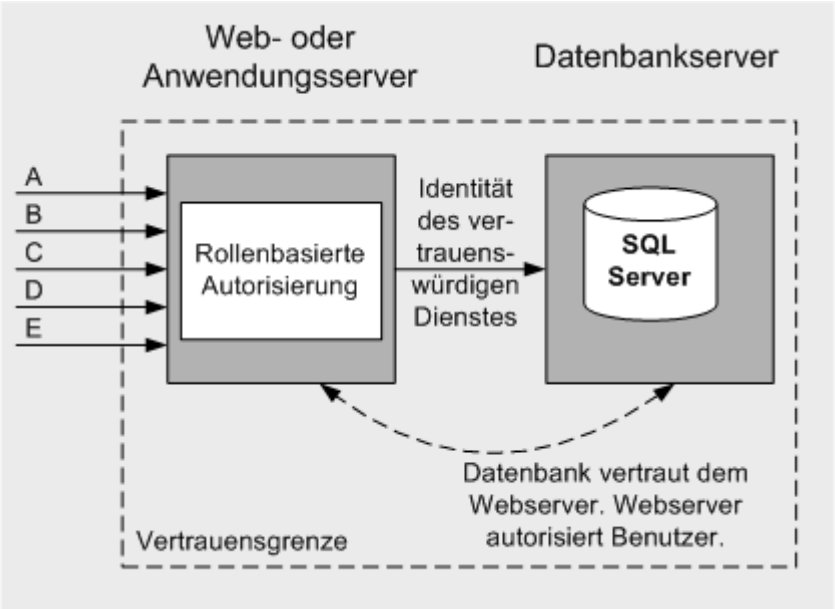

Abbildung 3.1
Das Modell mit vertrauenswürdigen Subsystemen

Der Ressourcenzugriff im Modell mit vertrauenswürdigen Subsystemen erfolgt folgendermaßen:

- Authentifizieren der Benutzer
- Zuordnen der Benutzer zu Rollen
- Autorisieren anhand der Rollenmitgliedschaft
- Zugriff auf die nachgeschaltete Ressourcenverwaltung mit einer festen vertrauten Identität

Feste Identitäten

Die feste Identität, mit der auf nachgeschaltete Systeme und Ressourcenverwaltungen zugegriffen wird, wird oftmals durch ein vorkonfiguriertes Windows-Konto bereitgestellt, das als Systemkonto bezeichnet wird. Bei einer Microsoft SQL Server™-Ressourcenverwaltung impliziert dies eine Windows-Authentifizierung gegenüber SQL Server.

Einige Anwendungen verwenden alternativ ein benanntes SQL-Konto (das in einer Verbindungszeichenfolge durch Benutzername und Kennwort angegeben wird) für den Zugriff auf SQL Server. In diesem Szenario muss die Datenbank für die SQL-Authentifizierung konfiguriert sein.

Weitere Informationen zu den relativen Vorzügen der Windows- und SQL-Authentifizierung bei der Kommunikation mit SQL Server finden Sie in Kapitel 12, "Datenzugriffssicherheit".

Verwenden mehrerer vertrauenswürdiger Identitäten

Einige Ressourcenverwaltungen müssen u. U. in der Lage sein, basierend auf der Rollenmitgliedschaft des Aufrufers eine etwas feinstufigere Autorisierung durchzuführen. Beispiel: Es gibt zwei Benutzergruppen, von denen eine autorisiert sein soll, Lese-/Schreiboperationen auszuführen, und die andere lediglich autorisiert sein soll, Leseoperationen auszuführen.

SQL Server bietet die folgende Möglichkeit:

- Erstellen Sie zwei Windows-Konten, eines für Leseoperationen und eines für Lese-/Schreiboperationen.

  Allgemeiner ausgedrückt: Es gibt zwei getrennte Konten, die die anwendungsspezifischen Rollen widerspiegeln. Sie können beispielsweise ein Konto für Internetbenutzer und das andere für interne Bediener und/oder Administratoren verwenden.

- Ordnen Sie jedem Konto eine benutzerdefinierte Datenbankrolle in SQL Server zu, und richten Sie die erforderlichen Datenbankberechtigungen für die einzelnen Rollen ein.
- Ordnen Sie die Benutzer in der Anwendung den Rollen zu, und ermitteln Sie anhand der Rollenmitgliedschaft, welches Konto einen Identitätswechsel durchführen muss, bevor die Verbindung mit der Datenbank hergestellt wird.

Diese Vorgehensweise ist in Abbildung 3.2 dargestellt.

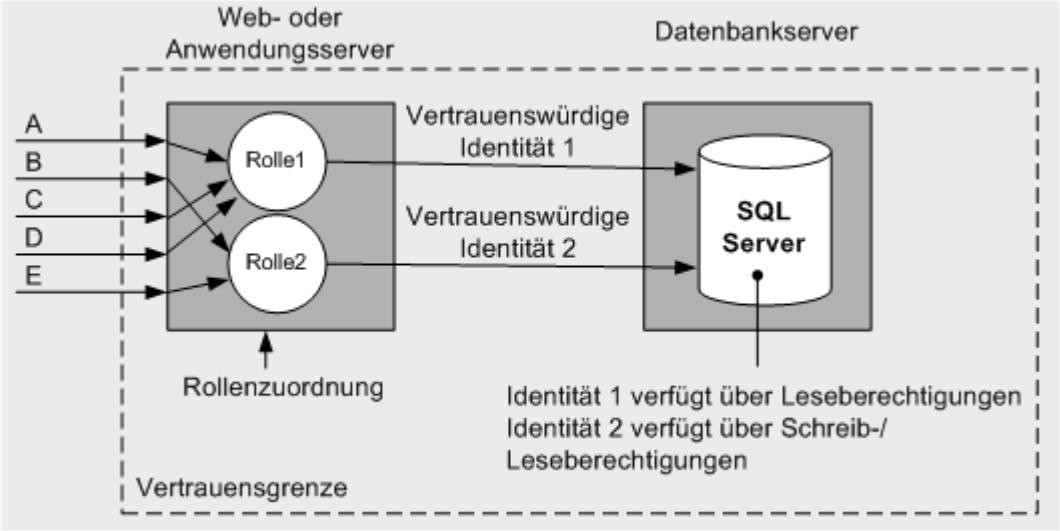

Abbildung 3.2
Verwenden mehrerer Identitäten für den Zugriff auf eine Datenbank, um eine feinstufigere Autorisierung zu unterstützen

Das Modell mit Identitätswechsel/Delegierung

Bei diesem Modell wechselt ein Dienst oder eine Komponente (normalerweise irgendwo in der logischen Unternehmensdienstebene) vor dem Zugriff auf den nächsten nachgeschalteten Dienst die Identität des Clients (durch einen Identitätswechsel auf Betriebssystemebene). Wenn sich der nachfolgende Dienst auf demselben Computer befindet, reicht ein Identitätswechsel aus. Wenn sich der nachgeschaltete Dienst auf einem Remotecomputer befindet, ist eine Delegierung erforderlich.

Die Delegierung führt dazu, dass der Sicherheitskontext für den Zugriff auf die nachgeschaltete Ressource mit dem des Clients übereinstimmt. Dieses Modell wird aus einer Reihe von Gründen meist verwendet:

- Es ermöglicht dem nachgeschalteten Dienst eine Autorisierung einzelner Benutzer anhand der Identität des ursprünglichen Benutzers.
- Es ermöglicht dem nachgeschalteten Dienst, Überwachungsfeatures auf Betriebssystemebene zu verwenden.

Ein konkretes Beispiel für diese Technik: Eine Enterprise Services-Komponente auf der mittleren Ebene wechselt die Identität des Aufrufers vor dem Zugriff auf eine Datenbank. Der Zugriff auf die Datenbank erfolgt unter Verwendung einer Datenbankverbindung, die an den Sicherheitskontext des ursprünglichen Aufrufers gebunden ist. Bei diesem Modell authentifiziert die Datenbank jeden Aufrufer. Sie trifft außerdem anhand der Berechtigungen, die der Identität des einzelnen Aufrufers zugewiesen wurden (bzw. der Windows-Gruppenmitgliedschaft des Aufrufers), die Entscheidungen zur Autorisierung. Das Modell mit Identitätswechsel/Delegierung ist in Abbildung 3.3 dargestellt.

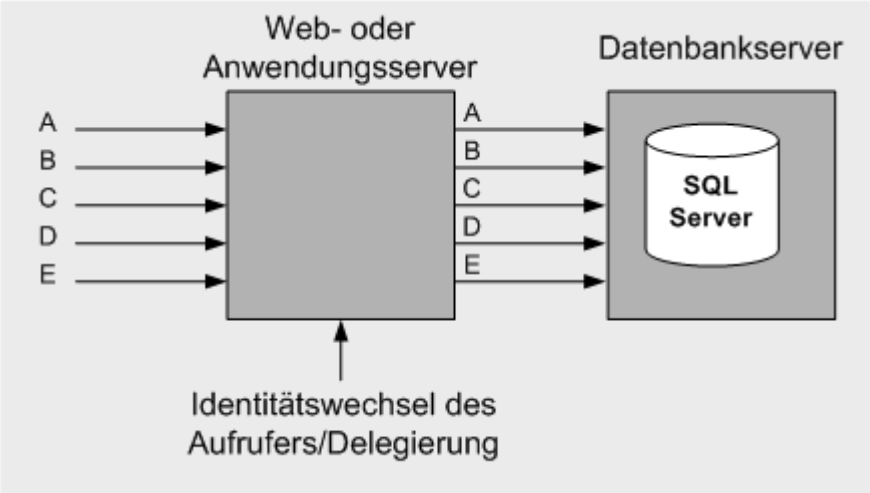

Abbildung 3.3
Das Modell mit Identitätswechsel/Delegierung

Auswählen eines Modells für den Ressourcenzugriff

Im Großteil der Internetanwendungen und weiträumigen Intranetanwendungen wird aus Gründen der Skalierbarkeit hauptsächlich das Modell mit vertrauenswürdigen Subsystemen verwendet. Das Modell mit Identitätswechsel wird in der Regel in kleineren Anwendungen verwendet, bei denen die Skalierbarkeit ohne große Bedeutung ist, sowie in Anwendungen, bei denen die Überwachung (aus Gründen der Nachweisbarkeit) unabdingbar ist.

Vorteil des Modells mit Identitätswechsel/Delegierung

Der Hauptvorteil des Modells mit Identitätswechsel/Delegierung ist die Überwachung (in der Nähe der Daten). Die Überwachung ermöglicht es Administratoren, nachzuverfolgen, welche Benutzer versucht haben, auf bestimmte Ressourcen zuzugreifen. Die Überwachung wird in der Regel dann als besonders maßgebend angesehen, wenn die Überwachungseinträge mit der genauen Zeit des Ressourcenzugriffs und von denselben Routinen generiert werden, die auf die Ressource zugreifen.

Dies wird vom Modell mit Identitätswechsel/Delegierung dadurch unterstützt, das der Sicherheitskontext für den Zugriff auf nachgeschaltete Ressourcen erhalten bleibt. Das Back-End-System ist somit in der Lage, den Benutzer und den angeforderten Zugriff maßgebend zu protokollieren.

Nachteile des Modells mit Identitätswechsel/Delegierung

Das Modell mit Identitätswechsel/Delegierung weist die folgenden Nachteile auf:

- **Technologische Schwierigkeiten** - Die meisten Sicherheitsdienstanbieter unterstützen (mit Kerberos als einzige Ausnahme) keine Delegierung.
  Prozesse, die einen Identitätswechsel durchführen, benötigen höhere Rechte (insbesondere das Recht **Als Teil des Betriebssystems handeln**). (Diese Einschränkung betrifft Windows 2000 und nicht Windows Server 2003).
- **Skalierbarkeit** - Das Modell mit Identitätswechsel/Delegierung bedingt, dass das Datenbankverbindungspooling nicht effektiv verwendet werden kann, da der Datenbankzugriff mithilfe von Verbindungen erfolgt, die an den jeweiligen Sicherheitskontext der ursprünglichen Aufrufer gebunden sind. Dadurch wird die Skalierungsmöglichkeit der Anwendung für eine Vielzahl von Benutzern erheblich eingeschränkt.
- **Erhöhter Verwaltungsaufwand** - Die ACLs der Back-End-Ressourcen müssen so verwaltet werden, dass jedem Benutzer die entsprechende Zugriffsebene gewährt wird. Wenn die Zahl der Back-End-Ressource (und die Anzahl der Benutzer) steigt, entsteht ein erheblicher Verwaltungsaufwand für die ACLs.

Vorteile des Modells mit vertrauenswürdigen Subsystemen

Das Modell mit vertrauenswürdigen Subsystemen bietet folgende Vorteile:

- **Skalierbarkeit** - Das Modell mit vertrauenswürdigen Subsystemen unterstützt das Verbindungspooling, eine wesentliche Voraussetzung für die Skalierbarkeit der Anwendung. Das Verbindungspooling ermöglicht es, dass mehrere Clients die verfügbaren gepoolten Verbindungen verwenden. Dies funktioniert bei diesem Modell, da bei jedem Zugriff auf Back-End-Ressourcen unabhängig von der Identität des Aufrufers der Sicherheitskontext des Systemkontos verwendet wird.
- **Minimieren der ACL-Verwaltung auf den Back-Ends** - Auf die Back-End-Ressourcen (z. B. Datenbanken) greift nur das Systemkonto zu. Die ACLs müssen nur für diese Identität konfiguriert werden.
- **Kein direkter Datenzugriff durch die Benutzer** - Im Modell mit vertrauenswürdigen Subsystemen wird lediglich dem Dienstkonto der mittleren Ebene Zugriff auf die Back-End-Ressourcen gewährt. Die Benutzer können daher nicht direkt auf die Back-End-Daten zugreifen, ohne die Anwendung zu durchlaufen (und von der Anwendung autorisiert zu werden).

Nachteile des Modells mit vertrauenswürdigen Subsystemen

Das Modell mit vertrauenswürdigen Subsystemen weist einige Nachteile auf:

- **Überwachung** - Um eine Überwachung im Back-End durchzuführen, kann die Identität des ursprünglichen Aufrufers (auf Anwendungsebene) explizit an das Back-End übergeben und die Überwachung dort durchgeführt werden. Dabei muss der mittleren Ebene vertraut werden, und es besteht das Risiko einer möglichen Nichtanerkennung. Sie können alternativ in der mittleren Ebene eine Überwachungsliste erzeugen und mit den Back-End-Überwachungslisten korrelieren (dazu muss sichergestellt sein, dass die Serveruhren synchronisiert sind).
- **Erhöhtes Risiko durch Serverkompromiss** - Im Modell mit vertrauenswürdigen Subsystemen wird dem Dienstkonto der mittleren Ebene weitreichender Zugriff auf die Back-End-Ressourcen gewährt. Somit erleichtert es ein gefährdeter Dienst auf der mittleren Ebene einem Angreifer, weitreichenden Zugriff auf Back-End-Ressourcen zu erhalten.

Identitätsübermittlung

Verteilte Anwendungen können in mehrere sichere Subsysteme unterteilt werden. Beispielsweise stellen eine Front-End-Webanwendung, ein Webdienst auf der mittleren Ebene, eine Remotekomponente und eine Datenbank vier verschiedene Sicherheitssubsysteme dar, die jeweils eine Authentifizierung und Autorisierung durchführen.

Sie müssen diejenigen Subsysteme identifizieren, durch die die Identität des Aufrufers (und der dazugehörige Sicherheitskontext) an das nächste nachgeschaltete Subsystem übermittelt werden muss, damit eine Autorisierung des ursprünglichen Aufrufers erfolgt.

Identitätsübermittlung auf Anwendungs- und Betriebssystemebene

Zu den Strategien für die Identitätsübermittlung zählen die Verwendung der Delegierungsfeatures des Betriebssystems oder die Übergabe von Tickets und/oder Anmeldeinformationen auf Anwendungsebene. Beispiel:

- Wenn die Identität auf Anwendungsebene übermittelt werden soll, werden die Anmeldeinformationen (oder Tickets) normalerweise mit Methodenargumenten oder Parametern gespeicherter Prozeduren übergeben.

Hinweis: GenericPrincipal-Objekte, die die Identität des authentifizierten Aufrufers übertragen, werden nicht automatisch zwischen Prozessen übermittelt. Dazu ist benutzerdefinierter Code erforderlich.

Sie können Parameter an gespeicherte Prozeduren übergeben, die das Abrufen und Verarbeiten benutzerspezifischer Daten ermöglichen. Beispiel:

```
SELECT CreditLimit From Table Where UserName="Bob"
```

Dieses Verfahren wird bisweilen als Verfahren mit *vertrauenswürdigen Abfrageparametern* bezeichnet.

- Bei einer Identitätsübermittlung auf Betriebssystemebene ist eine erweiterte Form des Identitätswechsel erforderlich, die sogenannte Delegierung.

Identitätswechsel und Delegierung

Unter bestimmten Umständen werden Threads in einer Serveranwendung unter dem Sicherheitskontext des Serverprozesses ausgeführt. Die Attribute, aus denen der Sicherheitskontext des Prozesses besteht, wird von der Anmeldesitzung des Prozesses verwaltet und vom Windows-Zugriffstoken auf Prozessebene offengelegt. Jeder lokale oder Remoteressourcenzugriff erfolgt mit dem Sicherheitskontext auf Prozessebene, der durch das Windows-Konto bestimmt wird, unter dem der Serverprozess ausgeführt wird.

Identitätswechsel

Wenn eine Serveranwendung für Identitätswechsel konfiguriert ist, wird dem Thread, mit dem eine Anforderung verarbeitet wird, ein Identitätswechseltoken zugeordnet. Das Identitätswechseltoken stellt den Sicherheitskontext des authentifizierten Aufrufers (bzw. des anonymen Benutzers) dar. Jeder lokale Ressourcenzugriff wird mit dem Identitätswechseltoken des Threads durchgeführt, sodass der Sicherheitskontext des Aufrufers verwendet wird.

Delegierung

Wenn der Thread der Serveranwendung versucht, auf eine Remoteressource zuzugreifen, ist eine Delegierung erforderlich. Insbesondere muss das Token des Aufrufers mit gewechselter Identität über Netzwerkanmeldeinformationen verfügen. Andernfalls erfolgt der Zugriff auf Remoteressourcen als anonymer Benutzer (AUTORITÄT\ANONYMOUS-ANMELDUNG).

Eine Reihe von Faktoren bestimmen, ob ein Sicherheitskontext delegiert werden kann. Tabelle 3.1 zeigt für die verschiedenen IIS-Authentifizierungstypen jeweils, ob der Sicherheitskontext des authentifizierten Aufrufers delegiert werden kann.

Tabelle 3.1: *IIS-Authentifizierungstypen*

Authentifizierungstyp	Delegierung möglich	Hinweise
Anonyme Authentifizierung	Fallabhängig	Wenn das anonyme Konto (standardmäßig IUSR_COMPUTER) in IIS als lokales Konto konfiguriert ist, kann es nur dann delegiert werden, wenn der lokale (Web-)Server und der Remotecomputer identische lokale Konten aufweisen (mit identischen Benutzernamen und Kennwörtern). Falls es sich bei dem anonymen Konto um ein Domänenkonto handelt, kann es delegiert werden.
Standardauthentifizierung	Ja	Wenn die Standardauthentifizierung mit lokalen Konten verwendet wird, kann es delegiert werden, sofern die Konten auf dem lokalen und dem Remotecomputer identisch sind. Domänenkonten können ebenfalls delegiert werden.
Digestauthentifizierung	Nein	
Integrierte Windows-Authentifizierung	Fallabhängig	Die integrierte Windows-Authentifizierung führt (abhängig von der Version des Betriebssystems auf dem Client- und dem Servercomputer) zu einer NTLM- oder Kerberos-Authentifizierung. NTLM unterstützt keine Delegierung. Bei einer geeignet konfigurierten Umgebung unterstützt Kerberos eine Delegierung. Weitere Informationen finden Sie im Abschnitt "Referenz" dieses Handbuchs unter "Vorgehensweise: Implementieren der Kerberos-Delegierung unter Windows 2000".
Clientzertifikate	Fallabhängig	Eine Delegierung ist möglich, wenn die IIS-Zertifikatszuordnung verwendet wird und das Zertifikat einem lokalen Konto, das auf den Remotecomputer dupliziert wurde, oder einem Domänenkonto zugeordnet wird. Dies funktioniert, weil die Anmeldeinformationen für das zugeordnete Konto auf dem lokalen Server gespeichert sind und eine interaktive Anmeldesitzung (mit Netzwerkanmeldeinformationen) erstellt wird. Die Active Directory-Zertifikatszuordnung unterstützt keine Delegierung.

Wichtig: Die Kerberos-Delegierung unter Windows 2000 unterliegt keinen Einschränkungen. Mit anderen Worten: Ein Benutzer kann evtl. mehrere Netzwerkhops über mehrere Remotecomputer hinweg durchführen. Um dieses potenzielle Sicherheitsrisiko zu vermeiden, sollten Sie die Reichweite des Domänenkontos begrenzen, indem das Konto aus der Gruppe Domänen-Benutzer entfernt wird und das Konto nur für die Anmeldung bei bestimmten Computern verwendet werden darf.

Rollenbasierte Autorisierung

Die meisten .NET-Webanwendungen verwenden ein rollenbasiertes Autorisierungsverfahren. Sie müssen sich über die verschiedenen Rollentypen klar werden und diejenigen auswählen, die sich für das Anwendungsszenario am besten eignen. Sie haben die folgenden Möglichkeiten:

- .NET-Rollen
- Enterprise Services (COM+)-Rollen
- Benutzerdefinierte SQL Server-Datenbankrollen
- SQL Server-Anwendungsrollen

.NET-Rollen

.NET-Rollen sind extrem flexibel und drehen sich um die **IPrincipal**-Objekte, die die Liste der Rollen enthalten, zu denen eine authentifizierte Identität gehört. .NET-Rollen können in Webanwendungen, Webdiensten oder Remotekomponenten verwendet werden, die in ASP.NET gehostet sind (und auf die mit **HttpChannel** zugegriffen wird).

Die Autorisierung mit .NET-Rollen kann entweder deklarativ unter Verwendung von **PrincipalPermission**-Forderungen oder programmatisch im Code unter Verwendung der imperativen **PrincipalPermission**-Forderungen bzw. der **IPrincipal.IsInRole**-Methode durchgeführt werden.

.NET-Rollen mit Windows-Authentifizierung

Wenn die Anwendung die Windows-Authentifizierung verwendet, erstellt ASP.NET automatisch einen **WindowsPrincipal**, der dem Kontext der aktuellen Webanforderung zugeordnet wird (mit **HttpContext.User**). Nachdem der Authentifizierungsvorgang abgeschlossen wurde und ASP.NET das Objekt der aktuellen Anforderung zugeordnet hat, wird das Objekt bei jeder nachfolgenden .NET-rollenbasierten Autorisierung verwendet.

Anhand der Windows-Gruppenmitgliedschaft des authentifizierten Aufrufers wird die Menge der Rollen ermittelt. Bei der Windows-Authentifizierung stimmen die .NET-Rollen mit den Windows-Gruppen überein.

.NET-Rollen ohne Windows-Authentifizierung

Wenn die Anwendung einen anderen Authentifizierungsmechanismus verwendet als die Windows-Authentifizierung, wie z. B. die Formularauthentifizierung oder die Passport-Authentifizierung, müssen Sie Code erstellen, um ein **GenericPrincipal**-Objekt (oder ein benutzerdefiniertes **IPrincipal**-Objekt) zu erstellen und darin die Menge der Rollen eintragen, die aus einem benutzerdefinierten Authentifizierungsdatenspeicher abgerufen wurden, z. B aus einer SQL Server-Datenbank.

Benutzerdefinierte IPrincipal-Objekte

Der .NET-rollenbasierte Sicherheitsmechanismus ist erweiterbar. Sie können eigene Klassen entwickeln, die **IPrincipal** und **IIdentity** implementieren und eine eigene erweiterte rollenbasierte Autorisierungsfunktionalität bereitstellen.

Solange das benutzerdefinierte **IPrincipal**-Objekt (mit den Rollen, die aus einem benutzerdefinierten Datenspeicher abgerufen wurden) dem Kontext der aktuellen Anforderung zugeordnet ist (mit **HttpContext.User**), ist die grundlegende Funktionalität der Rollenüberprüfung sichergestellt.

Durch Implementieren der **IPrincipal**-Schnittstelle wird sichergestellt, dass die deklarativen und die imperativen Formen von **PrincipalPermission**-Forderungen mit der benutzerdefinierten Identität funktionieren. Sie können darüber hinaus eine erweiterte Rollensemantik implementieren, indem Sie beispielsweise eine zusätzliche Methode wie **IsInMultipleRoles(string [] roles)** bereitstellen, die das Überprüfen und Bestätigen der Mitgliedschaft mehrerer Rollen ermöglichen.

Weitere Informationen

- Weitere Informationen zur .NET-rollenbasierten Autorisierung finden Sie in Kapitel 8, "ASP.NET-Sicherheit".
- Weitere Informationen zum Erstellen von **GenericPrincipal**-Objekten finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit GenericPrincipal-Objekten" im Abschnitt "Referenz" dieses Handbuchs.

Enterprise Services (COM+)-Rollen

Bei Verwendung von Enterprise Services (COM+)-Rollen werden die Zugriffsprüfungen in die mittlere Ebene verlagert, sodass beim Verbinden mit Back-End-Datenbanken ein Datenbankverbindungspooling verwendet werden kann. Um allerdings eine sinnvolle Enterprise Services (COM+)-rollenbasierte Autorisierung zu erreichen, muss die Front-End-Webanwendung einen Identitätswechsel durchführen und die Identität des ursprünglichen Aufrufers (mit einem Windows-Zugriffstoken) an die Enterprise Services-Anwendung weiterleiten. Dazu müssen die folgenden Einträge in die Datei **Web.config** der Webanwendung eingefügt werden.

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

Falls die Verwendung von deklarativen Prüfungen auf Methodenebene ausreicht (um zu ermitteln, welche Benutzer welche Methoden aufrufen können), können Sie die Anwendung bereitstellen und die Rollenmitgliedschaft mit dem Verwaltungsprogramm für Komponentendienste aktualisieren.

Wenn Im Code der Methoden programmatische Prüfungen erforderlich sind, gehen einige Vorteile bei der Verwaltung und Weitergabe von Enterprise Services (COM+)-Rollen verloren, da die Rollenlogik fest codiert ist.

Benutzerdefinierte SQL Server-Datenbankrollen

Bei diesem Verfahren werden die Rollen in der Datenbank erstellt, die Berechtigungen basierend auf den Rollen zugewiesen und den Rollen Windows-Gruppen- und -Benutzerkonten zugeordnet. Bei diesem Verfahren muss die Identität des Aufrufers bis zum Back-End gelangen (wenn Sie die bevorzugte Windows-Authentifizierung bei SQL Server verwenden).

SQL Server-Anwendungsrollen

Bei diesem Verfahren werden die Berechtigungen den Rollen in der Datenbank erteilt. SQL Server-Anwendungsrollen umfassen allerdings keine Benutzer- oder Gruppenkonten. Daher geht die Granularität des ursprünglichen Aufrufers verloren.

Bei Anwendungsrollen wird der Zugriff für eine bestimmte Anwendung autorisiert (und nicht für eine Gruppe von Benutzern). Die Anwendung aktiviert die Rolle mithilfe einer integrierten gespeicherten Prozedur, die einen Rollennamen und ein Kennwort übernimmt. Einer der Hauptnachteile dieses Verfahrens besteht darin, dass die Anwendung die Anmeldeinformationen (Rollenname und zugehöriges Kennwort) sicher verwalten muss.

Weitere Informationen

Weitere Informationen zu benutzerdefinierten SQL Server-Datenbankrollen und Anwendungsrollen finden Sie in Kapitel 12, "Datenzugriffssicherheit".

.NET-Rollen und Enterprise Services (COM+)-Rollen

Die folgende Tabelle enthält einen Vergleich der Features von .NET-Rollen und Enterprise Services (COM+)-Rollen.

Tabelle 3.2: *Vergleich von Enterprise Services-Rollen mit .NET-Rollen*

Feature	Enterprise Services-Rollen	.NET-Rollen
Verwaltung	Verwaltungsprogramm für Komponentendienste	Benutzerdefiniert
Datenspeicher	COM+-Katalog	Benutzerdefinierter Datenspeicher (z. B. SQL Server oder Active Directory)
Deklarativ	Ja [SecurityRole("Manager")]	Ja [PrincipalPermission(SecurityAction.Demand, Role="Manager")]
Imperativ	Ja ContextUtil.IsCallerInRole()	Ja IPrincipal.IsInRole
Granularität auf Klassen-, Schnittstellen- und Methoden-ebene	Ja	Ja
Erweiterbar	Nein	Ja (mithilfe einer benutzerdefinierten **IPrincipal**-Implementierung)
Verfügbar für alle .NET-Komponenten	Nur für Komponenten, die von der **ServicedComponent**-Basisklasse abstammen.	Ja
Rollenmit-gliedschaft	Rollen enthalten Windows-Gruppen- oder -Benutzerkonten	Bei Verwendung von **WindowsPrincipals** SIND Rollen Windows-Gruppen (keine zusätzliche Abstraktionsebene).
Explizite Schnittstellen-implemen-tierung erforderlich	Ja Für eine Autorisierung auf Methodenebene muss explizit eine Schnittstelle definiert und implementiert werden.	Nein

Verwenden von .NET-Rollen

Mit .NET-Rollen können die folgenden Elemente gesichert werden:

- Dateien
- Ordner
- Webseiten (ASPX-Dateien)
- Webdienste (ASMX-Dateien)
- Objekte
- Methoden und Eigenschaften
- Codeblöcke in Methoden

Die Tatsache, dass Sie mit .NET-Rollen Operationen (die von Methoden und Eigenschaften ausgeführt werden) und bestimmte Codeblöcke schützen können, bedeutet, dass der Zugriff auf lokale und Remoteressourcen geschützt werden kann, auf die die Anwendung zugreift.

Hinweis: Die ersten vier Elemente in der obigen Liste (Dateien, Ordner, Webseiten und Webdienste) werden mit **UrlAuthorizationModule** geschützt, das die Rollenmitgliedschaft des Aufrufers (und die Identität des Aufrufers) für Autorisierungsentscheidungen verwenden kann.

Wenn Sie die Windows-Authentifizierung verwenden, wird ein Großteil der erforderlichen Arbeit für die Verwendung von .NET-Rollen automatisch erledigt. ASP.NET erstellt ein **WindowsPrincipal**-Objekt, und die Windows-Gruppenmitgliedschaft des Benutzers bestimmt die zugehörigen Rollen.

Wenn Sie .NET-Rollen mit einem anderen Authentifizierungsmechanismus verwenden als der Windows-Authentifizierung, müssen Sie Code für die folgenden Aufgaben erstellen:

- Erfassen der Anmeldeinformationen des Benutzers.
- Überprüfen der Anmeldeinformationen des Benutzers anhand eines benutzerdefinierten Datenspeichers, z. B. einer SQL Server-Datenbank.
- Abrufen einer Rollenliste, Erstellen eines **GenericPrincipal**-Objekts und Zuordnen des Objekts zu der aktuellen Webanforderung.
  Das **GenericPrincipal**-Objekt stellt den authentifizierten Benutzer dar und wird für nachfolgende Überprüfungen der .NET-Rollen verwendet, z. B. deklarative **PrincipalPermission**-Forderungen und programmatische **IPrincipal.IsInRole**-Überprüfungen.

Weitere Informationen

Weitere Informationen zur Vorgehensweise beim Erstellen eines **GenericPrincipal**-Objekts für die Formularauthentifizierung finden Sie in Kapitel 8, "ASP.NET-Sicherheit".

Überprüfen der Rollenmitgliedschaft

Für die Überprüfung von .NET-Rollen gibt es die folgenden Möglichkeiten:

Wichtig: Die Überprüfung von .NET-Rollen setzt ein **IPrincipal**-Objekt voraus (das den authentifizierten Benutzer darstellt), das der aktuellen Anforderung zugeordnet ist. Bei ASP.NET-Webanwendungen muss das **IPrincipal**-Objekt **HttpContext.User** zugeordnet sein. Bei Windows Forms-Anwendungen muss das **IPrincipal**-Objekt **Thread.CurrentPrincipal** zugeordnet sein.

- **Manuelle Rollenprüfungen** - Für eine feinstufige Autorisierung können Sie die **IPrincipal.IsInRole**-Methode aufrufen, um den Zugriff auf bestimmte Codeblöcke anhand der Rollenmitgliedschaft des Aufrufers zu autorisieren. Beim Überprüfen der Rollenmitgliedschaft kann eine AND- und OR-Logik verwendet werden.
- **Deklarative Rollenprüfungen (Gates zu den Methoden)** - Sie können Methoden mit der **PrincipalPermissionAttribute**-Klasse versehen (die **PrincipalPermission** abgekürzt werden kann), um deklarativ eine Rollenmitgliedschaft zu fordern. Dabei wird nur die OR-Logik unterstützt. Sie können beispielsweise fordern, dass sich ein Aufrufer in mindestens einer bestimmten Rolle befindet (z B. Kassierer oder Manager). Sie können bei deklarativen Prüfungen nicht angeben, dass der Aufrufer Manager und Kassierer sein muss.
- **Imperative Rollenprüfungen (Prüfungen innerhalb der Methoden)** - Sie können im Code **PrincipalPermission.Demand** aufrufen, um eine feinstufige Autorisierungslogik auszuführen. Es werden logische AND- und OR-Operationen unterstützt.

Beispiele zur Rollenprüfung

Die folgenden Codefragmente zeigen einige Beispiele für Rollenprüfungen mit programmatischen, deklarativen und imperativen Techniken.

Autorisieren von Bob, eine Operation durchzuführen:

Hinweis: Obwohl Sie einzelne Benutzer autorisieren können, sollten Sie die Autorisierung in der Regel anhand der Rollenmitgliedschaft durchführen, bei der Sie Gruppen von Benutzern mit denselben Rechten in der Anwendung autorisieren können.

- Direkte Überprüfung des Benutzernamens

```
GenericIdentity userIdentity = new GenericIdentity("Bob");
if (userIdentity.Name=="Bob")
{
}
```

- Deklarative Prüfung

```
[PrincipalPermissionAttribute(SecurityAction.Demand, User="Bob")]
public void DoPrivilegedMethod()
{
}
```

- Imperative Prüfung

```
PrincipalPermission permCheckUser = new PrincipalPermission(
                                                "Bob", null);
permCheckUser.Demand();
```

Autorisieren von Kassierern (Teller), eine Operation durchzuführen:

- Direkte Überprüfung des Rollennamens

```
GenericIdentity userIdentity = new GenericIdentity("Bob");
// Role names would be retrieved from a custom data store
string[] roles = new String[]{"Manager", "Teller"};
GenericPrincipal userPrincipal = new GenericPrincipal(userIdentity,
                                                      roles);
if (userPrincipal.IsInRole("Teller"))
{
}
```

- Deklarative Prüfung

```
[PrincipalPermissionAttribute(SecurityAction.Demand, Role="Teller")]
void SomeTellerOnlyMethod()
{
}
```

- Imperative Prüfung

```
public SomeMethod()
{
  PrincipalPermission permCheck = new PrincipalPermission(
                                                null,"Teller");
  permCheck.Demand();
  // Only Tellers can execute the following code
  // Non members of the Teller role result in a security exception
  . . .
}
```

Autorisieren von Managern oder (OR) Kassierern (Teller), eine Operation durchzuführen:

- Direkte Überprüfung des Rollennamens

```
if (Thread.CurrentPrincipal.IsInRole("Teller") ||
    Thread.CurrentPrincipal.IsInRole("Manager"))
{
  // Perform privileged operations
}
```

- Deklarative Prüfung

```
[PrincipalPermissionAttribute(SecurityAction.Demand, Role="Teller"),
 PrincipalPermissionAttribute(SecurityAction.Demand, Role="Manager")]
public void DoPrivilegedMethod()
{
…
}
```

- Imperative Prüfung

```
PrincipalPermission permCheckTellers = new PrincipalPermission(
                                                  null,"Teller");
PrincipalPermission permCheckManagers = new PrincipalPermission(
                                                  null,"Manager");
(permCheckTellers.Union(permCheckManagers)).Demand();
```

Autorisieren nur der Personen, die Manager und (AND) Kassierer (Teller) sind, eine Operation durchzuführen:

- Direkte Überprüfung des Rollennamens

```
if (Thread.CurrentPrincipal.IsInRole("Teller") &&
    Thread.CurrentPrincipal.IsInRole("Manager"))
{
  // Perform privileged operation
}
```

- Deklarative Prüfung
  Es ist nicht möglich, bei .NET-Rollen deklarativ AND-Prüfungen durchzuführen. Die Stapelung von **PrincipalPermission**-Forderungen führt zu einem logischen OR.
- Imperative Prüfung

```
PrincipalPermission permCheckTellers = new PrincipalPermission(
                                                 null,"Teller");
permCheckTellers.Demand();
PrincipalPermission permCheckManagers = new PrincipalPermission(
                                                 null, "Manager");
permCheckManagers.Demand();
```

Auswählen eines Authentifizierungsmechanismus

Dieser Abschnitt enthält Informationen, die Ihnen bei der Auswahl eines geeigneten Authentifizierungsmechanismus für gebräuchliche Anwendungsszenarien helfen sollen. Sie sollten zunächst die folgenden Aspekte betrachten:

- **Identitäten** - Ein Windows-Authentifizierungsmechanismus eignet sich nur dann, wenn die Benutzer der Anwendung über Windows-Konten verfügen, die von einer vertrauenswürdigen Autorität authentifiziert werden können, die vom Webserver der Anwendung erreichbar ist.
- **Verwalten von Anmeldeinformationen** - Ein Hauptvorteil der Windows-Authentifizierung besteht darin, dass Sie die Verwaltung der Anmeldeinformationen dem Betriebssystem überlassen können. Bei anderen Verfahren als der Windows-Authentifizierung, z. B. bei der Formularauthentifizierung, müssen Sie sorgfältig überlegen, wo und wie die Anmeldeinformationen der Benutzer gespeichert werden sollen. Am häufigsten werden die folgenden Verfahren verwendet:
  - SQL Server-Datenbanken
  - Benutzerobjekte in Active Directory

  Weitere Informationen zu Sicherheitsaspekten bei Verwendung von SQL Server als Speicher für die Anmeldeinformationen finden Sie in Kapitel 12, "Datenzugriffssicherheit".

  Weitere Informationen zur Verwendung der Formularauthentifizierung mit benutzerdefinierten Datenspeichern (einschließlich Active Directory) finden Sie in Kapitel 8, "ASP.NET-Sicherheit".
- **Identitätsübermittlung** - Müssen Sie ein Modell mit Identitätswechsel/Delegierung implementieren und den Sicherheitskontext des ursprünglichen Aufrufers auf Betriebssystemebene durch die Ebenen übertragen, um beispielsweise eine Überwachung oder eine (feinstufige) Autorisierung einzelner Benutzer zu unterstützen? In diesem Fall müssen Sie in der Lage sein, die Identität des Aufrufers zu wechseln und dessen Sicherheitskontext an das nächste nachgeschaltete Subsystem zu delegieren (siehe Abschnitt "Delegierung" weiter oben in diesem Kapitel).
- **Browsertyp** - Verfügen alle Benutzer über Internet Explorer, oder muss eine Benutzerbasis mit gemischten Browsertypen unterstützt werden? Tabelle 3.3 veranschaulicht, welche Authentifizierungsmechanismen den Browser Internet Explorer erfordern und welche eine Vielzahl gebräuchlicher Browsertypen unterstützen.

  Tabelle 3.3: *Browseranforderungen im Hinblick auf die Authentifizierung*

Authentifizierungstyp	Internet Explorer erforderlich	Hinweise
Formularauthentifizierung	Nein	
Passport-Authentifizierung	Nein	
Integrierte Windows-Authentifizierung (Kerberos oder NTLM)	Ja	Kerberos erfordert außerdem das Betriebssystem Windows 2000 oder höher auf den Client- und Servercomputern sowie Konten, die für eine Delegierung konfiguriert sind. Weitere Informationen finden Sie unter "Vorgehensweise: Implementieren der Kerberos-Delegierung unter Windows 2000" im Abschnitt "Referenz" dieses Handbuchs.
Standardauthentifizierung	Nein	Die Standardauthentifizierung ist Bestandteil des Protokolls HTTP 1.1, das nahezu von allen Browsern unterstützt wird.
Digestauthentifizierung	Ja	
Zertifikatauthentifizierung	Nein	Die Clients benötigen X.509-Zertifikate.

Internetszenarien

- Bei Internetszenarien gelten die folgenden Grundannahmen:
- Die Benutzer verfügen über keine Windows-Konten in der Domäne des Servers bzw. in einer vertrauten Domäne, die vom Server erreichbar ist.
- Die Benutzer verfügen über keine Clientzertifikate.

Abbildung 3.4 zeigt eine Entscheidungsstruktur für die Auswahl eines Authentifizierungsmechanismus für Internetszenarien.

Abbildung 3.4
Auswahl eines Authentifizierungsmechanismus für Internetanwendungen

Weitere Informationen zur Webdienstsicherheit und die Spezifikation der Webdienstsicherheit (WS-Security), die Bestandteil der GXA-Initiative (Global XML Architecture) ist, finden Sie in Kapitel 10, "Webdienstsicherheit".

Vergleich von Formularauthentifizierung und Passport-Authentifizierung

In diesem Abschnitt finden Sie eine Übersicht über die relativen Vorzüge der Formularauthentifizierung und der Passport-Authentifizierung.

Vorteile der Formularauthentifizierung

- Unterstützt die Authentifizierung anhand eines benutzerdefinierten Datenspeichers, normalerweise eine SQL Server-Datenbank oder Active Directory.
- Unterstützt die rollenbasierte Autorisierung mit Suche der Rollen in einem Datenspeicher.
- Nahtlose Integration in die Webbenutzeroberfläche.
- ASP.NET stellt einen Großteil der Infrastruktur bereit. Im Vergleich zum klassischen ASP ist relativ wenig benutzerdefinierter Code erforderlich.

Vorteile der Passport-Authentifizierung

- Passport ist eine zentralisierte Lösung.
- Die Anwendung wird von der Verwaltung der Anmeldeinformationen entbunden.
- Die Passport-Authentifizierung kann bei rollenbasierten Autorisierungsschemas verwendet werden.
- Die Passport-Authentifizierung ist extrem sicher, da sie auf Kryptographietechnologien aufbaut.

Weitere Informationen

- Weitere Informationen zu Authentifizierungsverfahren für Webdienste finden Sie in Kapitel 10, "Webdienstsicherheit".
- Weitere Informationen zum Verwenden der Formularauthentifizierung mit SQL Server finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit SQL Server 2000" im Abschnitt "Referenz" dieses Handbuchs.

Intranet-/Extranetszenarien

Abbildung 3.5 zeigt eine Entscheidungsstruktur, die Ihnen bei der Auswahl eines Authentifizierungsmechanismus für Intranet- und Extranetanwendungsszenarien behilflich sein kann.

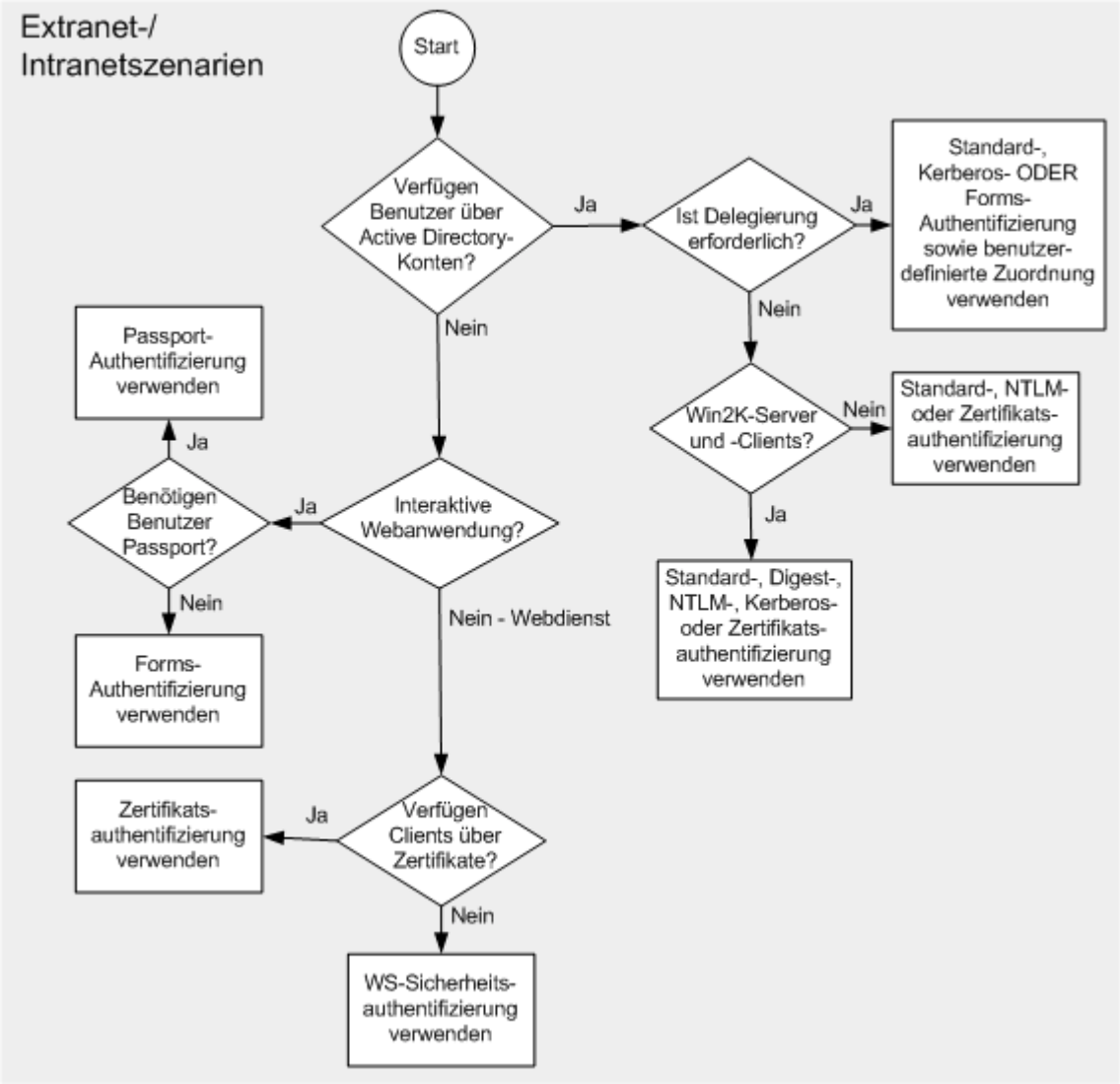

Abbildung 3.5
Auswahl eines Authentifizierungsmechanismus für Intranet- und Extranetanwendungen

Vergleich der Authentifizierungsmechanismen

Die folgende Tabelle enthält einen Vergleich der verfügbaren Authentifizierungsmechanismen.

Tabelle 3.4: *Verfügbare Authentifizierungsmethoden*

	Standard	Digest	NTLM	Kerberos	Zertifikate	Formular	Passport
Die Benutzer benötigen Windows-Konten in der Domäne des Servers	Ja	Ja	Ja	Ja	Nein	Nein	Nein
Delegierung wird unterstützt*	Ja	Nein	Nein	Ja	Ja	Ja	Ja
Erfordert Win2K-Clients und –Server	Nein	Ja	Nein	Ja	Nein	Nein	Nein
Anmeldeinformationen werden unverschlüsselt übertragen (SSL erforderlich)	Ja	Nein	Nein	Nein	Nein	Ja	Nein
Unterstützung anderer Browser als IE	Ja	Nein	Nein	Nein	Ja	Ja	Ja

* Weitere Informationen finden Sie im Abschnitt "Identitätsübermittlung" weiter oben in diesem Kapitel unter "Delegierung".

Zusammenfassung

- Der Entwurf von Authentifizierungs- und Autorisierungsverfahren für verteilte Anwendungen stellt eine anspruchsvolle Aufgabe dar. Ein einwandfreier Authentifizierungs- und Autorisierungsentwurf in den frühen Entwurfsphasen der Anwendungsentwicklung hilft dabei, viele häufige Sicherheitsrisiken zu mildern. Nachstehend finden Sie eine Zusammenfassung der Informationen in diesem Kapitel:
- Verwenden Sie das Ressourcenzugriffsmodell mit vertrauenswürdigen Subsystemen, um die Vorteile eines Datenbankverbindungspoolings zu nutzen.
- Verwenden Sie die .NET-Rollenprüfung, wenn die Anwendung keine Windows-Authentifizierung verwendet, um die Autorisierung bereitzustellen. Überprüfen Sie die Anmeldeinformationen anhand eines benutzerdefinierten Datenspeichers, rufen Sie eine Liste der Rollen ab, und erstellen Sie ein **GenericPrincipal**-Objekt. Ordnen Sie das Objekt der aktuellen Webanforderung zu (**HttpContext.User**).
- Verwenden Sie .NET-Rollen, wenn die Anwendung die Windows-Authentifizierung, aber keine Enterprise Services verwendet. Beachten Sie, dass .NET-Rollen bei der Windows-Authentifizierung Windows-Gruppen sind.
- Sie sollten die Verwendung von Enterprise Services (COM+)-Rollen erwägen, wenn die Anwendung die Windows-Authentifizierung und Enterprise Services verwendet.
- Für eine sinnvolle rollenbasierte Autorisierung mit Enterprise Services (COM+)-Rollen muss die Identität des ursprünglichen Aufrufers an die Enterprise Services-Anwendung übergeben werden. Wenn die Enterprise Services-Anwendung von einer ASP.NET-Webanwendung aufgerufen wird, bedeutet das, dass die Webanwendung die Windows-Authentifizierung verwenden und für Identitätswechsel konfiguriert sein muss.
- Versehen Sie Methoden mit dem **PrincipalPermission**-Attribut, um deklarativ eine Rollenmitgliedschaft zu fordern. Die Methode wird nicht aufgerufen, wenn sich der Aufrufer nicht in der angegebenen Rolle befindet. In diesem Fall wird eine Sicherheitsausnahme generiert.
- Rufen Sie für feinstufige Autorisierungsentscheidungen **PrincipalPermission.Demand** im Methodencode auf (oder verwenden Sie **IPrincipal.IsInRole**).
- Erwägen Sie die Implementierung eines benutzerdefinierten **IPrincipal**-Objekts, wenn Sie eine zusätzliche Semantik für die Rollenprüfung erreichen möchten.