Erstellen sicherer ASP.NET-Anwendungen

Authentifizierung, Autorisierung und sichere Kommunikation

Kapitel 2 – Sicherheitsmodell für ASP.NET-Anwendungen

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 beschreibt die allgemeinen Merkmale von .NET-Webanwendungen im Hinblick auf die Sicherheit und stellt das Sicherheitsmodell von .NET-Webanwendungen vor. Darüber hinaus werden die Kernimplementierungstechnologien vorgestellt, mit denen sichere .NET-Webanwendungen erstellt werden.

Inhalt

In diesem Kapitel wird die Sicherheit von .NET-Webanwendungen behandelt. Es enthält eine Übersicht über die Sicherheitsfeatures und –dienste, die in allen Ebenen einer normalen .NET-Webanwendung vorhanden sind.

Das Kapitel verfolgt die folgenden Ziele:

- Bereitstellen eines Referenzrahmens für normale .NET-Webanwendungen.
- Identifizieren der Sicherheitsfeatures für Authentifizierung, Autorisierung und sichere Kommunikation, die von den verschiedenen Implementierungstechnologien bereitgestellt werden, mit denen .NET-Webanwendungen erstellt werden.
- Identifizieren der Gatekeeper und Gates, mit denen in der Anwendung Vertrauensgrenzen erzwungen werden können.

.NET-Webanwendungen

Dieser Abschnitt enthält eine kurze Einführung in .NET-Webanwendungen und beschreibt deren Merkmale aus logischer und physischer Sicht. Darüber hinaus werden die verschiedenen Implementierungstechnologien vorgestellt, mit denen .NET-Webanwendungen erstellt werden.

Logische Ebenen

Die logische Anwendungsarchitektur zeigt ein System als Gruppe kooperierender Dienste, die in den folgenden Schichten gruppiert sind:

- Benutzerdienste
- Unternehmensdienste
- Datendienste

Der Wert dieser logischen Architekturansicht besteht darin, die generischen Diensttypen zu identifizieren, die gleich bleibend in jedem System vorhanden sind, um eine passende Segmentierung sicherzustellen und die Schnittstellen zwischen den Ebenen zu definieren. Diese Segmentierung ermöglicht es Ihnen, beim Implementieren der einzelnen Schichten Entscheidungen im Hinblick auf Architektur und Entwurf umsichtiger zu treffen und eine besser verwaltbare Anwendung zu erstellen.

Die Schichten können folgendermaßen beschrieben werden:

- **Die Benutzerdienste** sind für die Interaktion des Clients mit dem System verantwortlich und stellen eine gemeinsame Brücke in die Kerngeschäftslogik bereit, die durch Komponenten in der Unternehmensdienstschicht gekapselt ist. Die Benutzerdienste sind seit jeher oftmals interaktiven Benutzern zugeordnet. Sie führen jedoch auch die erste Verarbeitung von programmatischen Anforderungen anderer Systeme durch, bei denen keine sichtbare Benutzeroberfläche involviert ist. Authentifizierung und Autorisierung, deren Art jeweils vom jeweiligen Clienttyp abhängt, werden normalerweise in der Benutzerdienstschicht durchgeführt.
- **Die Unternehmensdienste** stellen die Kernfunktionalität des Systems bereit und kapseln die Geschäftslogik. Sie sind unabhängig von dem Übermittlungskanal und Back-End-Systemen oder Datenquellen. Dadurch wird die erforderliche Stabilität und Flexibilität bereitgestellt, damit das System später ggf. neue und andere Kanäle und Back-End-Systeme unterstützen kann. An der Verarbeitung einer bestimmten Geschäftsanforderung sind in der Regel mehrere kooperierende Komponenten in der Unternehmensdienstschicht beteiligt.
- **Die Datendienste** ermöglichen über generische Schnittstellen, die von den Komponenten in der Unternehmensdienstschicht verwendet werden können, den Zugriff auf Daten, die innerhalb der Systemgrenzen gespeichert sind, und auf andere (Back-End-)Systeme. Die Datendienste abstrahieren die Vielfältigkeit der Back-End-Systeme und Datenquellen und kapseln bestimmte Zugriffsregeln und Datenformate.

Die logische Klassifikation der Diensttypen in einem System kann mit der möglichen physischen Verteilung der Komponenten, die die Dienste implementieren, korrelieren, ist aber relativ unabhängig davon.

Sie sollten außerdem bedenken, dass die logischen Ebenen auf jeder Aggregationsstufe identifiziert werden können. Die Ebenen können also für das System als Ganzes (im Kontext der entsprechenden Umgebung und den externen Interaktionen) und für jedes enthaltene Subsystem identifiziert werden. Jeder Remoteknoten, der einen Webdienst hostet, besteht aus Benutzerdiensten (die eingehende Anforderungen und Nachrichten behandeln), Unternehmensdiensten und Datendiensten.

Physische Bereitstellungsmodelle

Die oben beschriebenen logischen Dienstschichten implizieren in keiner Weise eine bestimmte Anzahl physischer Ebenen. Alle drei logischen Dienste können sich physisch auf demselben Computer befinden oder auf mehrere Computer verteilt sein.

Der Webserver als Anwendungsserver

Ein allgemeines Bereitstellungsmuster für .NET-Webanwendungen besteht darin, die Geschäfts- und Datenzugriffskomponenten auf dem Webserver zu speichern. Dadurch werden die Hops im Netzwerk reduziert, wodurch die Systemleistung steigen kann. Dieses Modell ist in Abbildung 2.1 dargestellt.

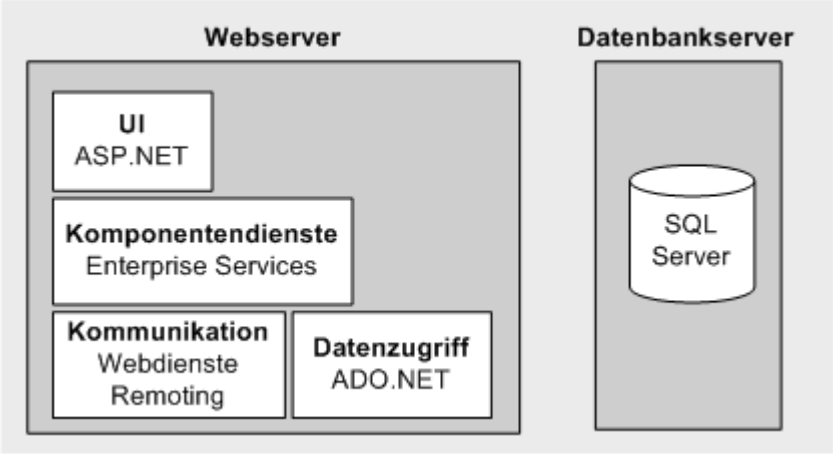

Abbildung 2.1
Der Webserver als Anwendungsserver

Remoteanwendungsebene

Die Remoteanwendungsebene ist ein gebräuchliches Bereitstellungsmuster, insbesondere für Internetszenarien, bei denen die Webschicht in einem Perimeternetzwerk (auch DMZ, demilitarisierte Zone, und abgeschirmtes Subnetz genannt) enthalten und durch Firewalls mit Paketfiltern von den Endbenutzern und der Remoteanwendungsebene getrennt ist. Die Remoteanwendungsebene ist in Abbildung 2.2 dargestellt.

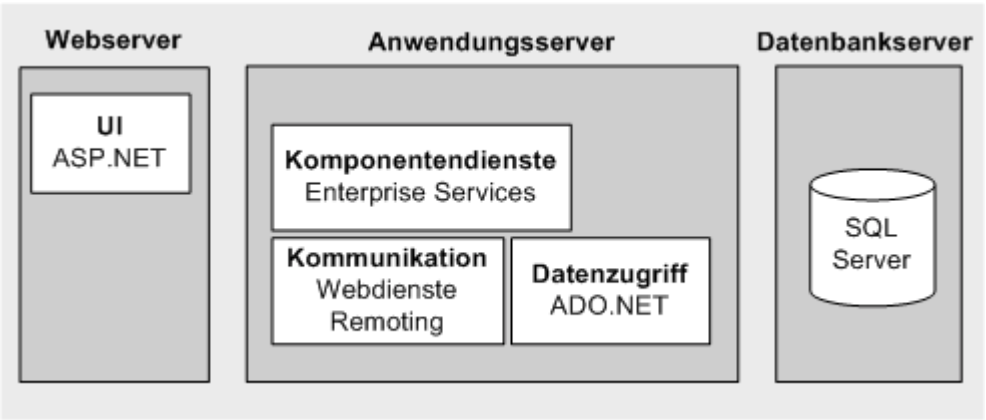

Abbildung 2.2
Einfügen einer Remoteanwendungsebene

Implementierungstechnologien

.NET-Webanwendungen implementieren normalerweise mithilfe der folgenden Technologien einen oder mehrere der logischen Dienste:

- ASP.NET
- Enterprise Services
- Webdienste
- .NET Remoting
- ADO.NET und Microsoft® SQL Server™ 2000
- IPSec (Internet Protocol Security)
- Secure Sockets Layer (SSL)

ASP.NET

Mit ASP.NET werden normalerweise die Benutzerdienste implementiert. ASP.NET stellt eine austauschbare Architektur bereit, mit der Webseiten erstellt werden können. Weitere Informationen zu ASP.NET finden Sie in den folgenden Ressourcen:

- Kapitel 8, "ASP.NET Sicherheit“
- "ASP.NET" im Abschnitt "Referenzliste" dieses Handbuchs

Enterprise Services

Enterprise Services stellen für Anwendungen Dienste auf Infrastrukturebene bereit. Dazu zählen Dienste für verteilte Transaktionen und Ressourcenverwaltung, wie z. B. Objektpooling für .NET-Komponenten. Weitere Informationen zu Enterprise Services finden Sie in den folgenden Ressourcen:

- Kapitel 9, "Enterprise Services-Sicherheit"
- "Understanding Enterprise Services (COM+) in .NET" in MSDN (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/entserv.asp) (nur auf Englisch verfügbar).
- "Enterprise Services" im Abschnitt "Referenzliste" dieses Handbuchs

Webdienste

Webdienste ermöglichen den Austausch von Daten und den Remoteaufruf der Anwendung mithilfe eines SOAP-basierten Nachrichtenaustauschs, um Daten über Firewalls und zwischen heterogenen Systemen zu übertragen. Weitere Informationen zu Webdiensten finden Sie in den folgenden Ressourcen:

- Kapitel 10, "Webdienstsicherheit"
- "XML Web Services Development Center" in MSDN (http://msdn.microsoft.com/webservices/) (nur auf Englisch verfügbar).
- "Webdienste" im Abschnitt "Referenzliste" dieses Handbuchs

.NET Remoting

.NET Remoting stellt ein Framework für den prozess- und computerübergreifenden Zugriff auf verteilte Objekte bereit. Weitere Informationen zu .NET Remoting finden Sie in den folgenden Ressourcen:

- Kapitel 11, ".NET Remoting-Sicherheit"
- ".NET Remoting" im Abschnitt "Referenzliste" dieses Handbuchs

ADO.NET und SQL Server 2000

ADO.NET stellt Dienste für den Datenzugriff bereit und wurde von Grund auf für verteilte Webanwendungen entworfen. ADO.NET enthält eine umfassende Unterstützung für Szenarien mit Trennung der Verbindung, die systembedingt bei Webanwendungen vorkommen. Weitere Informationen zu ADO.NET finden Sie in den folgenden Ressourcen:

- Kapitel 12, "Datenzugriffssicherheit"
- "ADO.NET" im Abschnitt "Referenzliste" dieses Handbuchs

SQL Server bietet eine integrierte Sicherheit, die die Authentifizierungsmechanismen (Kerberos oder NTLM) des Betriebssystems verwendet. Die Autorisierung wird durch Anmeldenamen und detaillierte Berechtigungen bereitgestellt, die auf einzelne Datenbankobjekte angewendet werden können. Weitere Informationen zu SQL Server 2000 finden Sie in den folgenden Ressourcen:

- Kapitel 12, "Datenzugriffssicherheit"

IPSec (Internet Protocol Security)

IPSec stellt durchgehende Verschlüsselungs- und Authentifizierungsdienste auf Transportebene bereit. Weitere Informationen zu IPSec finden Sie in den folgenden Ressourcen:

- Kapitel 4, "Sichere Kommunikation"
- *IPSec – The New Security Standard for the Internet, Intranets and Virtual Private Networks* von Naganand Doraswamy und Dan Harkins (Prentice Hall PTR, ISBN; ISBN: 0-13-011898-2); Kapitel 4 steht im TechNet zur Verfügung (http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/prodtech/network/ipsecarc.asp?frame=true) (nur auf Englisch verfügbar).

Secure Sockets Layer (SSL)

SSL stellt einen durchgehenden sicheren Kommunikationskanal bereit. Die über den Kanal übertragenen Daten sind verschlüsselt. Weitere Informationen zu SSL finden Sie in den folgenden Ressourcen:

- Kapitel 4, "Sichere Kommunikation"
- *Microsoft® Windows® 2000 and IIS 5.0 Administrator's Pocket Consultant* (Microsoft Press, ISBN: 0-7356-1024-X); Kapitel 6 steht im TechNet zur Verfügung (http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/iis/maintain/featusability/c06iis.asp) (nur auf Englisch verfügbar).

Sicherheitsarchitektur

Abbildung 2.3 zeigt das Modell der Remoteanwendungsebene zusammen mit den Sicherheitsdiensten, die von den oben vorgestellten Technologien bereitgestellt werden. Authentifizierung und Autorisierung werden in allen Schichten an mehreren einzelnen Punkten durchgeführt. Diese Dienste werden hauptsächlich durch die Internet-Informationsdienste (IIS), ASP.NET, Enterprise Services und SQL Server bereitgestellt. Alle Ebenen sind durch sichere Kommunikationskanäle verbunden, die sich vom Clientbrowser oder –gerät bis zur Datenbank erstrecken. Die Kanäle werden durch eine Kombination aus SSL (Secure Sockets Layer) oder IPSec gesichert.

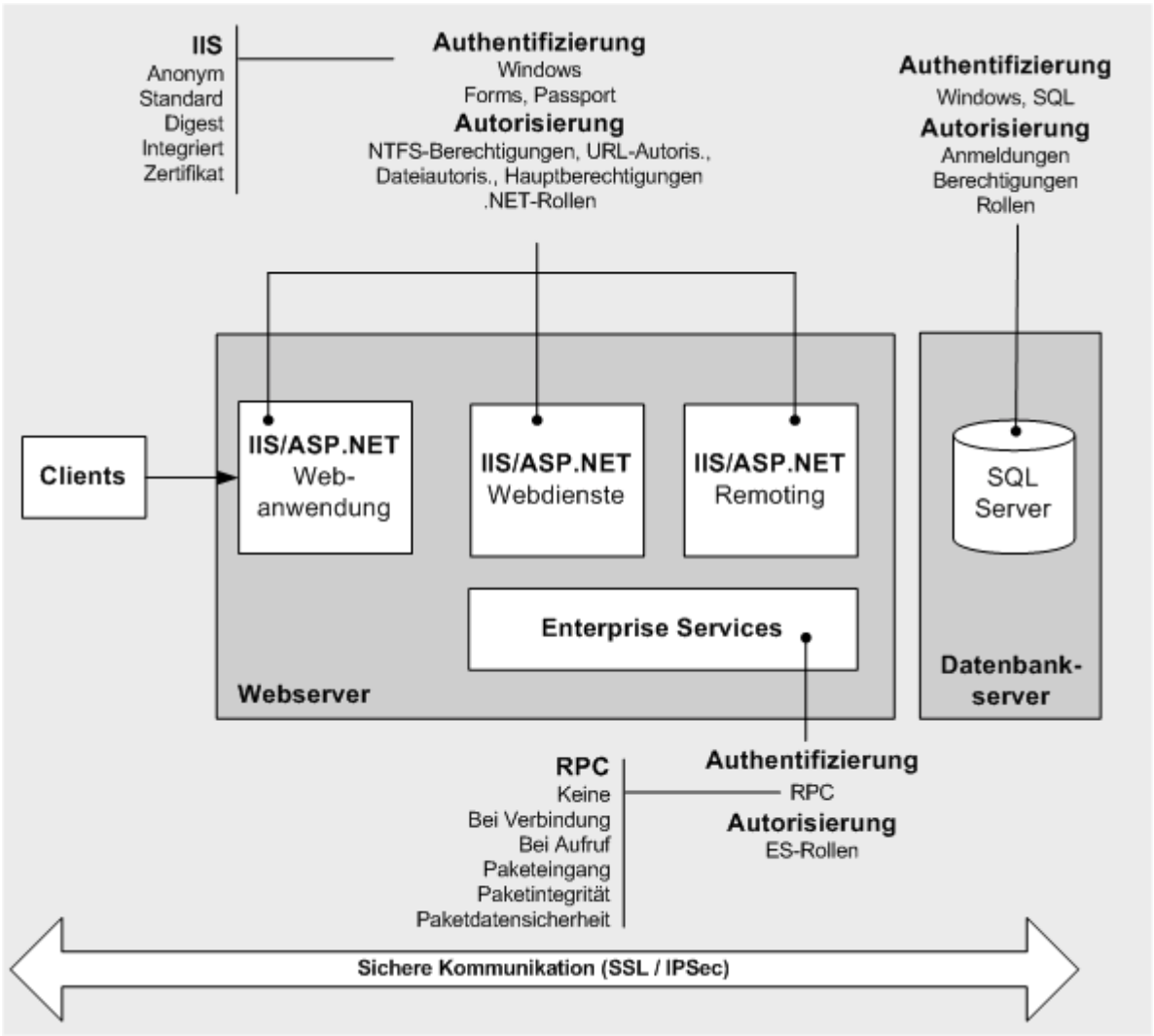

Abbildung 2.3
Sicherheitsarchitektur

Ebenenübergreifende Sicherheit

Die Features für Authentifizierung, Autorisierung und sichere Kommunikation, die von den oben beschriebenen Technologien bereitgestellt werden, sind in Tabelle 2.1 zusammengefasst.

Tabelle 2.1: *Sicherheitsfeatures*

Technologie	Authentifizierung	Autorisierung	Sichere Kommunikation
IIS	Anonyme Authentifizierung Standardauthentifizierung Digestauthentifizierung Integrierte Windows-Authentifizierung (Kerberos/NTLM) Zertifikatsauthentifizierung	IP/DNS-Adress-einschränkungen Webberechtigungen NTFS-Berechtigungen; Windows-Zugriffs-steuerungslisten für angeforderte Dateien	SSL
ASP.NET	Keine (Benutzerdefiniert) Windows-Authentifizierung Formularauthentifizierung Passport-Authentifizierung	Dateiautorisierung URL-Autorisierung Principalberechtigungen .NET-Rollen	
Webdienste	Windows-Authentifizierung Keine (Benutzerdefiniert) Authentifizierung auf Nachrichtenebene	Dateiautorisierung URL-Autorisierung Principalberechtigungen .NET-Rollen	SSL und Verschlüsselung auf Nachrichtenebene
Remoting	Windows-Authentifizierung	Dateiautorisierung URL-Autorisierung Principalberechtigungen .NET-Rollen	SSL und Verschlüsselung auf Nachrichtenebene
Enterprise Services	Windows-Authentifizierung	Enterprise Services (COM+)-Rollen NTFS-Berechtigungen	RPC-Verschlüsselung (Remote Procedure Call)
SQL Server 2000	Windows-Authentifizierung (Kerberos/NTLM) SQL-Authentifizierung	Serveranmeldenamen Datenbankanmeldenamen Feste Datenbankrollen Benutzerdefinierte Rollen Anwendungsrollen Objektberechtigungen	SSL
Windows 2000	Kerberos NTLM	Windows-ACLs	IPSec

Authentifizierung

Das .NET Framework in Windows 2000 stellt die folgenden Authentifizierungsoptionen bereit:

- ASP.NET-Authentifizierungsmodi
- Enterprise Services-Authentifizierung
- SQL Server-Authentifizierung

ASP.NET-Authentifizierungsmodi

Die ASP.NET-Authentifizierungsmodi umfassen Windows-Authentifizierung, Formularauthentifizierung, Passport-Authentifizierung und Keine Authentifizierung.

- **Windows-Authentifizierung –** Bei diesem Authentifizierungsmodus verlässt sich ASP.NET auf IIS, um Benutzer zu authentifizieren und ein Windows-Zugriffstoken zu erstellen, das die authentifizierte Identität darstellt. IIS stellt die folgenden Authentifizierungsmechanismen bereit:
  - **Standardauthentifizierung –** Bei der Standardauthentifizierung müssen die Benutzer Anmeldeinformationen in der Form von Benutzername und Kennwort eingeben, um ihre Identität zu beweisen. Hierbei handelt es sich um einen Vorschlag für einen Internetstandard basierend auf RFC 2617: http://www.faqs.org/rfcs/rfc2617.html. Sowohl Netscape Navigator als auch Microsoft Internet Explorer unterstützen die Standardauthentifizierung. Die Anmeldeinformationen des Benutzers werden vom Browser in einem unverschlüsselten Base64-codierten Format an den Webserver gesendet. Da der Webserver die Anmeldeinformationen des Benutzers unverschlüsselt empfängt, kann der Webserver Remoteaufrufe mit den Anmeldeinformationen des Benutzers ausführen (um beispielsweise auf Remotecomputer und –ressourcen zuzugreifen).

    Hinweis: Die Standardauthentifizierung sollte nur zusammen mit einem sicheren Kanal verwendet werden (der normalerweise mit SSL eingerichtet wird), da die Benutzernamen und Kennwörter mit einer Netzwerküberwachungssoftware leicht gestohlen werden können. Wenn Sie die Standardauthentifizierung verwenden, sollten Sie in allen Seiten (nicht nur auf einer Anmeldeseite) SSL verwenden, da die Anmeldeinformationen bei allen folgenden Anforderungen übergeben werden. Weitere Informationen zur Verwendung der Standardauthentifizierung mit SSL finden Sie in Kapitel 8, "ASP.NET-Sicherheit".

  - **Digestauthentifizierung –** Die Digestauthentifizierung, die mit IIS 5.0 eingeführt wurde, ähnelt der Standardauthentifizierung mit der Ausnahme, dass die Anmeldeinformationen des Benutzers nicht unverschlüsselt vom Browser an den Webserver, sondern ein Hash der Anmeldeinformationen gesendet wird. Diese Authentifizierung ist sicherer. Allerdings sind ein Internet Explorer 5.0-Client (oder höher) und eine spezielle Serverkonfiguration erforderlich.
  - **Integrierte Windows-Authentifizierung –** Die integrierte Windows-Authentifizierung (Kerberos oder NTLM, abhängig von der Client- und Serverkonfiguration) verwendet einen kryptografischen Austausch mit dem Internet Explorer-Webbrowser des Benutzers, um die Identität des Benutzers zu bestätigen. Sie wird nur von Internet Explorer (und nicht von Netscape Navigator) unterstützt. Sie wird daher in der Regel nur in Intranetszenarien verwendet, in denen die Clientsoftware kontrolliert werden kann. Sie wird vom Webserver nur verwendet, wenn der anonyme Zugriff deaktiviert ist bzw. der anonyme Zugriff durch Berechtigungen des Windows-Dateisystems verweigert wird.

- **Zertifikatsauthentifizierung –** Bei der Zertifikatsauthentifizierung werden Clientzertifikate verwendet, um Benutzer positiv zu identifizieren. Das Clientzertifikat wird vom Browser des Benutzers (oder der Clientanwendung) an den Webserver übergeben. (Bei Webdiensten übergibt der Webdienstclient das Zertifikat mithilfe der **ClientCertificates**-Eigenschaft des **HttpWebRequest**-Objekts). Anschließend extrahiert der Webserver die Identität des Benutzers aus dem Zertifikat. Dieses Verfahren setzt voraus, dass auf dem Computer des Benutzers ein Clientzertifikat installiert ist. Es wird daher in der Regel meistens in Intranet- oder Extranetszenarien verwendet, in denen die Anzahl der Benutzer bekannt und kontrolliert ist. Wenn IIS ein Clientzertifikat empfängt, kann das Zertifikat einem Windows-Konto zugeordnet werden.
- **Anonyme Authentifizierung –** Wenn die Clients nicht authentifiziert werden müssen oder ein benutzerdefiniertes Authentifizierungsschema implementiert wird, kann IIS für die anonyme Authentifizierung konfiguriert werden. In diesem Fall erstellt der Webserver ein Windows-Zugriffstoken, um alle anonymen Benutzer mit dem gleichen anonymen Konto (ggf. Gastkonto) darzustellen. Das anonyme Standardkonto ist IUSR_COMPUTERNAME, wobei COMPUTERNAME für den NetBIOS-Namen des Computers steht, der bei der Installation angegeben wurde.

- **Passport-Authentifizierung –** Bei diesem Authentifizierungsmodus verwendet ASP.NET die zentralisierten Authentifizierungsdienste von Microsoft Passport. ASP.NET stellt einen praktischen Wrapper um die Funktionalität des Microsoft Passport Software Development Kit (SDK) bereit, das auf dem Webserver installiert sein muss.
- **Formularauthentifizierung –** Bei diesem Verfahren wird eine clientseitige Umleitung verwendet, um nicht authentifizierte Benutzer an ein bestimmtes HTML-Formular umzuleiten, auf dem die Anmeldeinformationen eingegeben werden können (normalerweise Benutzername und Kennwort). Anschließend werden diese Anmeldeinformationen überprüft und ein Authentifizierungsticket generiert, das an den Client zurückgegeben wird. Das Authentifizierungsticket enthält die Identität des Benutzers und optional eine Liste der Rollen, denen der Benutzer für die Dauer der Benutzersitzung angehört.

  Die Formularauthentifizierung wird bisweilen nur für die Personalisierung von Websites verwendet. In diesem Fall muss im geringen Umfang benutzerdefinierter Code erstellt werden, da ASP.NET bei einfacher Konfiguration einen Großteil des Verfahrens automatisch erledigt. Bei Personalisierungsszenarien muss das Cookie nur den Benutzernamen enthalten.

Hinweis: Bei der Formularauthentifizierung werden Benutzername und Kennwort unverschlüsselt an den Webserver übertragen. Die Formularauthentifizierung sollte daher mit einem durch SSL gesicherten Kanal verwendet werden. Damit der Schutz des Authentifizierungscookies, das bei nachfolgenden Anforderungen übertragen wird, erhalten bleibt, sollten Sie erwägen, SSL für alle Seiten in der Anwendung zu verwenden, nicht nur für die Anmeldeseite.

- **Keine –** Keine Authentifizierung bedeutet, dass die Benutzer entweder nicht authentifiziert werden sollen oder ein benutzerdefiniertes Authentifizierungsprotokoll verwendet wird.

Weitere Informationen

Weitere Einzelheiten zur ASP.NET-Authentifizierung finden Sie in Kapitel 8, "ASP.NET-Sicherheit".

Enterprise Services-Authentifizierung

Die Enterprises Services-Authentifizierung wird mithilfe der zugrunde liegenden RPC-Transportinfrastruktur (Remote Procedure Call) durchgeführt, die wiederum die SSPI (Security Service Provider Interface) des Betriebssystems verwendet. Clients von Enterprise Services-Anwendungen können mit der Kerberos- oder der NTLM-Authentifizierung authentifiziert werden.

Eine Serviced Component kann in einer Bibliotheksanwendung oder in einer Serveranwendung gehostet werden. Bibliotheksanwendungen sind in Clientprozessen gehostet und nehmen daher die Identität des Clients an. Serveranwendungen werden in getrennten Serverprozessen mit eigener Identität ausgeführt. Weitere Informationen zur Identität finden Sie weiter unten in diesem Kapitel unter "Identitäten und Principals".

Die eingehenden Aufrufe an eine Serviced Component können auf den folgenden Ebenen authentifiziert werden:

- **Standard**: Verwendung der Standardauthentifizierungsebene für das Sicherheitspaket.
- **Keine**: Keine Authentifizierung.
- **Verbinden**: Die Authentifizierung findet nur statt, wenn die Verbindung hergestellt wird.
- **Aufruf**: Authentifizierung am Anfang jedes Remoteprozeduraufrufs.
- **Paket**: Authentifizierung und Bestätigung, dass alle Aufrufdaten empfangen wurden.
- **Paketintegrität**: Authentifizierung und Bestätigung, dass keine Daten bei der Übertragung verändert wurden.
- **Paketsicherheit**: Authentifizierung und Verschlüsselung des Pakets, einschließlich der Daten sowie der Identität und Signatur des Absenders.

Weitere Informationen

Weitere Einzelheiten zur Enterprise Services-Authentifizierung finden Sie in Kapitel 9, "Enterprise Services-Sicherheit".

SQL Server-Authentifizierung

SQL Server kann Benutzer mithilfe der Windows-Authentifizierung (NTLM oder Kerberos) authentifizieren oder das eigene integrierte Authentifizierungsschema, die SQL-Authentifizierung, verwenden. Die folgenden beiden Optionen stehen zur Verfügung:

- **SQL Server und Windows –** Clients können unter Verwendung der SQL Server-Authentifizierung oder der Windows-Authentifizierung eine Verbindung zu einer Instanz von Microsoft SQL Server herstellen. Dies wird bisweilen als Authentifizierung im gemischten Modus bezeichnet.
- **Nur Windows –** Der Benutzer muss unter Verwendung der Windows-Authentifizierung eine Verbindung zu einer Instanz von Microsoft SQL Server herstellen.

Weitere Informationen

Die relativen Vorteile der einzelnen Verfahren werden in Kapitel 12, "Datenzugriffssicherheit", erläutert.

Autorisierung

Das .NET Framework in Windows 2000 stellt die folgenden Autorisierungsoptionen bereit:

- ASP.NET-Autorisierungsoptionen
- Enterprise Services-Autorisierung
- SQL Server-Autorisierung

ASP.NET-Autorisierungsoptionen

Die ASP.NET-Autorisierungsoptionen können von ASP.NET-Webanwendungen, Webdiensten und Remotekomponenten verwendet werden. ASP.NET stellt die folgenden Autorisierungsoptionen bereit:

- **URL-Autorisierung –** Dieser Autorisierungsmechanismus wird durch Einstellungen in Computer- und Anwendungskonfigurationsdateien konfiguriert. Die URL-Autorisierung ermöglicht die Einschränkung des Zugriffs auf bestimmte Dateien und Ordner im URI-Namespace (Uniform Resource Identifier) der Anwendung. Sie können beispielsweise bestimmten Benutzern den Zugriff auf bestimmte Dateien oder Ordner (die durch einen URL adressiert werden) verweigern oder gestatten. Sie können den Zugriff auch anhand der Rollenmitgliedschaft des Benutzers sowie anhand des Typs der HTTP-Anforderung (GET, POST usw.) einschränken.

  Die URL-Autorisierung erfordert eine authentifizierte Identität, die durch ein Windows- oder ticketbasiertes Authentifizierungsschema erhalten werden kann.
- **Dateiautorisierung –** Die Dateiautorisierung trifft nur dann zu, wenn ein von IIS bereitgestellter Windows-Authentifizierungsmechanismus verwendet wird, um Aufrufer zu authentifizieren, und wenn ASP.NET für die Windows-Authentifizierung konfiguriert ist.

  Sie können damit den Zugriff auf bestimmte Dateien auf einem Webserver einschränken. Die Zugriffsberechtigungen werden durch Windows-ACLs bestimmt, die den Dateien zugeordnet sind.
- **PrincipalPermission-Forderungen –** PrincipalPermission-Forderungen können (deklarativ oder programmatisch) als zusätzlicher feinstufiger Zugriffssteuerungsmechanismus verwendet werden. Sie ermöglichen basierend auf der Identität und Gruppenmitgliedschaft einzelner Benutzer die Steuerung des Zugriffs auf Klassen, Methoden oder einzelne Codeblöcke.
- **.NET-Rollen –** Mit .NET-Rollen werden Benutzer mit den gleichen Berechtigungen in der Anwendung gruppiert. Vom Konzept her ähneln sie früheren rollenbasierten Implementierungen, z. B. Windows-Gruppen und COM+-Rollen. Im Gegensatz zu diesen früheren Verfahren erfordern .NET-Rollen keine authentifizierten Windows-Identitäten und können mit ticketbasierten Authentifizierungsschemas wie der Formularauthentifizierung verwendet werden.

  Mit .NET-Rollen kann der Zugriff auf Ressourcen und Operationen gesteuert werden. Sie können deklarativ und programmatisch konfiguriert werden.

Weitere Informationen

Weitere Einzelheiten zur ASP.NET-Autorisierung finden Sie in Kapitel 8, "ASP.NET-Sicherheit".Enterprise Services-Autorisierung

Der Zugriff auf die Funktionalität von Serviced Components in Enterprise Services-Anwendungen wird durch die Enterprise Services-Rollenmitgliedschaft geregelt. Diese unterscheiden sich von den .NET-Rollen und können Windows-Gruppen- oder -Benutzerkonten enthalten. Die Rollenmitgliedschaft ist im COM+-Katalog definiert und wird mit dem Tool für Komponentendienste verwaltet.

Weitere Informationen

Weitere Einzelheiten zur Enterprise Services-Autorisierung finden Sie in Kapitel 9, "Enterprise Services-Sicherheit".

SQL Server-Autorisierung

SQL Server ermöglicht feinstufige Berechtigungen, die auf einzelne Datenbankobjekte angewendet werden können. Die Berechtigungen können auf der Rollenmitgliedschaft basieren (SQL Server stellt feste Datenbankrollen, benutzerdefinierte Rollen und Anwendungsrollen bereit) oder können einzelnen Windows-Benutzer- oder –Gruppenkonten gewährt werden.

Weitere Informationen

Weitere Einzelheiten zur SQL Server-Autorisierung finden Sie in Kapitel 12, "Datenzugriffssicherheit".

Gatekeeper und Gates

Im restlichen Teil dieses Dokuments wird mit dem Begriff *Gatekeeper* die Technologie bezeichnet, die für ein *Gate* verantwortlich ist. Ein Gate stellt einen Zugriffssteuerungspunkt (für eine Ressource) in einer Anwendung dar. Bei einer Ressource kann es sich beispielsweise um eine Operation (die durch eine Methode eines Objekts dargestellt wird) oder einen Datenbank- bzw. Dateisystemressource handeln.

Alle oben aufgeführten Kerntechnologien stellen Gatekeeper für die Zugriffsautorisierung bereit. Die Anforderungen müssen eine Reihe von Gates durchlaufen, bevor sie auf die angeforderte Ressource oder Operation zugreifen dürfen. Die Gates, die Anforderungen durchlaufen müssen, werden nachstehend beschrieben:

- IIS stellt ein Gate bereit, wenn die Benutzer authentifiziert werden (d. h., die anonyme Authentifizierung deaktiviert ist). Die IIS-Webberechtigungen können als Zugriffssteuerungsmechanismus verwendet werden, um die Möglichkeit der Webbenutzer einzuschränken, auf bestimmte Dateien und Ordner zuzugreifen. Im Gegensatz zu NTFS-Dateiberechtigungen gelten Webberechtigungen für alle Webbenutzer und nicht für einzelne Benutzer oder Gruppen. NTFS-Dateiberechtigungen ermöglichen weitere Einschränkungen für Webressourcen wie Webseiten, Bilddateien usw. Diese Einschränkungen gelten für einzelne Benutzer oder Gruppen.

  IIS überprüft zuerst die Webberechtigungen und anschließend die NTFS-Dateiberechtigungen. Ein Benutzer muss durch beide Mechanismen autorisiert werden, damit er auf die Datei bzw. den Ordner zugreifen kann. Wenn die Überprüfung der Webberechtigung fehlschlägt, gibt IIS die Antwort „HTTP 403 – Zugriff verboten“ zurück. Wenn die Überprüfung der NTFS-Berechtigung fehlschlägt, gibt IIS „HTTP 401 – Zugriff verweigert“ zurück.
- ASP.NET stellt verschiedene konfigurierbare und programmatische Gates bereit. Dazu zählen URL-Autorisierung, Dateiautorisierung, PrincipalPermission-Forderungen und .NET-Rollen.
- Die Enterprise Services-Gatekeeper verwenden Enterprise Services-Rollen, um den Zugriff auf die Geschäftsfunktionalität zu autorisieren.
- SQL Server 2000 enthält eine Reihe von Gates, die Serveranmeldenamen, Datenbankanmeldenamen und Datenbankobjektberechtigungen umfassen.
- Windows 2000 stellt Gates mithilfe von ACLs bereit, die sicheren Ressourcen zugeordnet sind.

Generell gilt, dass Gatekeeper die Autorisierung basierend auf der Identität des Benutzers oder Dienstes durchführen, der das Gate aufruft und versucht, auf eine bestimmte Ressource zuzugreifen. Der Wert mehrerer Gates liegt in der tief greifenden Sicherheit mit mehreren Verteidigungslinien. Tabelle 2.2 fasst die Gatekeeper zusammen und identifiziert jeweils die Gates, für die sie verantwortlich sind.

Tabelle 2.2: *Verantwortung der Gatekeeper und die jeweils bereitgestellten Gates*

Gatekeeper	Gates
Windows-Betriebssystem	Anmelderechte (positiv und negativ, z. B. **Lokale Anmeldung verweigern**) Sonstige Rechte (z. B. **Als Teil des Betriebssystems handeln**) Überprüfung des Zugriffs auf gesicherte Ressourcen wie Registrierung und Dateisystem. Die Zugriffsüberprüfungen verwenden ACLs, die den sicheren Ressourcen zugeordnet sind und die berechtigten Benutzer/Dienste, die auf die Ressource zugreifen dürfen, sowie die zulässigen Operationen angeben. TCP/IP-Filterung IP-Sicherheit
IIS	Authentifizierung (anonyme Authentifizierung, Standardauthentifizierung, Digestauthentifizierung, integrierte Windows-Authentifizierung und Zertifikatsauthentifizierung). Einschränkungen von IP-Adressen und Domänennamen. (Diese Einschränkungen können als zusätzliche Verteidigungslinie verwendet werden. Allerdings sollte sich darauf nicht verlassen werden, da IP-Adressen relativ leicht nachgeahmt werden können.) Webberechtigungen NTFS-Berechtigungen
ASP.NET	URL-Autorisierung Dateiautorisierung PrincipalPermission-Forderungen .NET-Rollen
Enterprise Services	Windows-Authentifizierung (NTLM / Kerberos) Enterprise Services (COM+)-Rollen Identitätswechselebenen
Webdienste	Verwendet die von IIS und ASP.NET bereitgestellten Gates.
Remoting	Verwendet die vom Host bereitgestellten Gates. Bei Hosting in ASP.NET werden die von IIS und ASP.NET bereitgestellten Gates verwendet. Bei Hosting in einem Windows-Dienst muss eine benutzerdefinierte Lösung entwickelt werden.
ADO.NET	Verbindungszeichenfolgen. Die Anmeldeinformationen können explizit sein. Sie können aber auch die Windows-Authentifizierung verwenden (beispielsweise beim Verbinden mit SQL Server).
SQL Server	Serveranmeldenamen Datenbankanmeldenamen Datenbankobjektberechtigungen

Mithilfe der verschiedenen Gates in allen Ebenen der Anwendung können die Benutzer ausgefiltert werden, denen der Zugriff auf die Back-End-Ressourcen gestattet werden soll. Der Gültigkeitsbereich des Zugriffs wird durch aufeinanderfolgende Gates eingeschränkt, die immer feinstufiger werden, desto weiter die Anforderung durch die Anwendung an die Back-End-Ressourcen gelangt.

Betrachten Sie dazu das internetbasierte Anwendungsbeispiel, in dem IIS verwendet wird (siehe Abbildung 2.4).

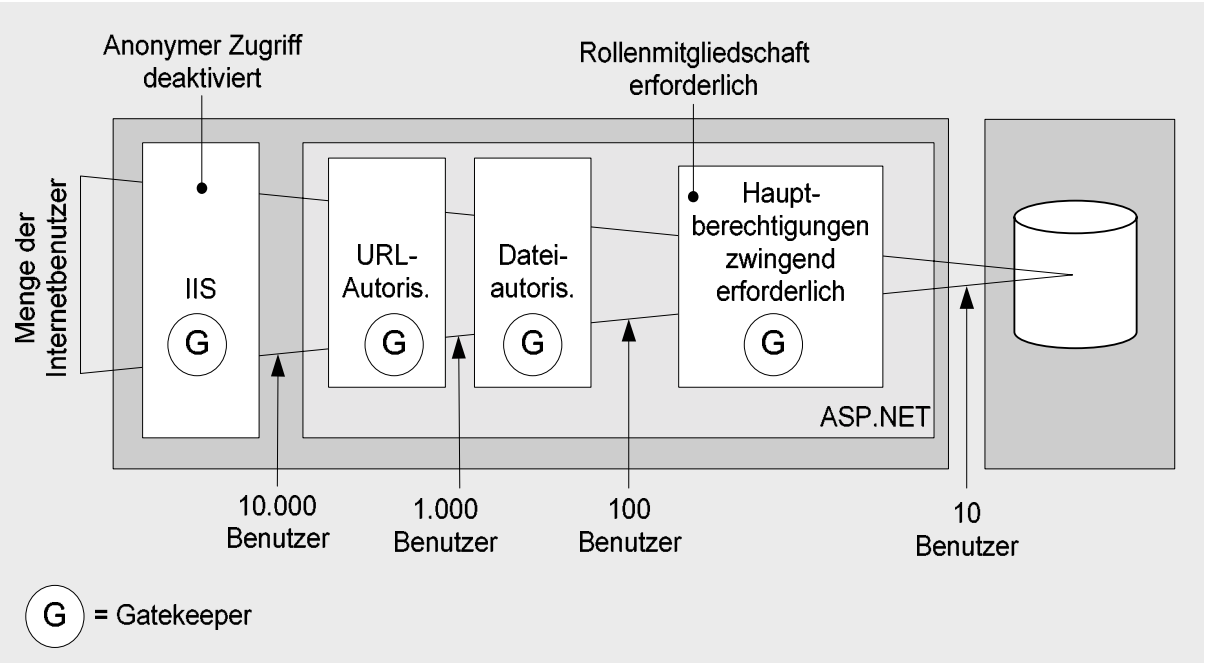

Abbildung 2.4
Filtern von Benutzern mit Gatekeepern

Abbildung 2.4 veranschaulicht Folgendes:

- Die anonyme Authentifizierung kann in IIS deaktiviert werden. Daher wird nur den Konten, die IIS authentifizieren kann, der Zugriff gestattet. Dadurch wird die mögliche Anzahl der Benutzer auf 10.000 reduziert.
- Als Nächstes wird in ASP.NET die URL-Autorisierung verwendet, die die Anzahl der Benutzer auf 1.000 reduzieren kann.
- Durch eine Dateiautorisierung kann der Zugriff weiter auf 100 Benutzer eingeschränkt werden.
- Schließlich kann der Webanwendungscode basierend auf einer bestimmten Rollenmitgliedschaft zulassen, dass lediglich 10 Benutzer auf die eingeschränkte Ressource zugreifen.

Identitäten und Principals

Die .NET-Sicherheit ist oberhalb der Windows-Sicherheit angesiedelt. Das benutzerzentrische Konzept der Windows-Sicherheit basiert auf einem Sicherheitskontext, der von einer Anmeldesitzung bereitgestellt wird, während die .NET-Sicherheit auf den Objekten **IPrincipal** und **IIdentity** basiert.

Wenn Sie bei der Windows-Programmierung feststellen möchten, unter welchen Sicherheitskontext der Code ausgeführt wird, wird die Identität des Prozessbesitzers bzw. des aktuell ausgeführten Threads herangezogen. Wenn Sie bei der .NET-Programmierung den Sicherheitskontext des aktuellen Benutzers abfragen möchten, wird das aktuelle **IPrincipal**-Objekt aus **Thread.CurrentPrincipal** abgerufen.

Das .NET Framework verwendet die Identitäts- und Principalobjekte, um Benutzer darzustellen, wenn .NET-Code ausgeführt wird. Sie bilden zusammen das Rückgrat der rollebasierten Sicherheit von .NET.

Die Identitäts- und Principalobjekte müssen die Schnittstellen **IIdentity** bzw. **IPrincipal** implementieren. Diese Schnittstellen sind im **System.Security.Principal**-Namespace definiert. Gemeinsame Schnittstellen ermöglichen es, dass Identitäts- und Principalobjekte im .NET Framework unabhängig von den zugrunde liegenden Implementierungsdetails polymorph behandelt werden.

Die **IPrincipal**-Schnittstelle ermöglicht die Überprüfung der Rollenmitgliedschaft mit der **IsInRole**-Methode und ermöglicht ebenfalls Zugriff auf das zugeordnete **IIdentity**-Objekt.

```
public interface IPrincipal
{
  bool IsInRole( string role );
  IIdentity Identity {get;}
}
```

Die **IIdentity**-Schnittstelle stellt zusätzliche Authentifizierungsdetails bereit, wie z. B. Name und Authentifizierungstyp.

```
public interface IIdentity
{
  string authenticationType {get;}
  bool IsAuthenticated {get;}
  string Name {get;}
}
```

Das .NET Framework stellt eine Reihe konkreter Implementierungen von **IPrincipal** und **IIdentity** bereit (siehe Abbildung 2.5 und Beschreibung in den folgenden Abschnitten).

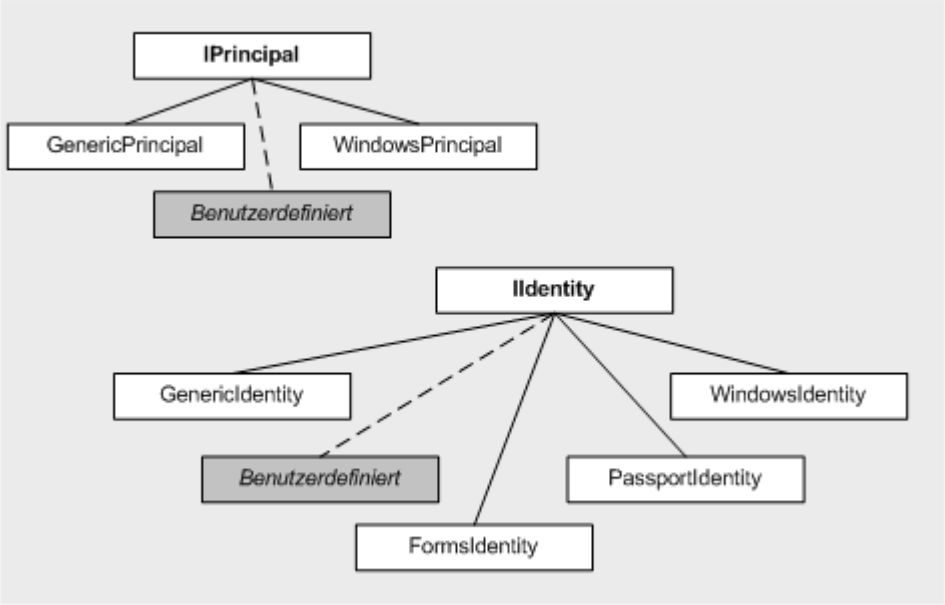

Abbildung 2.5
IPrincipal- und IIdentity-Implementierungsklassen

WindowsPrincipal und WindowsIdentity

Die .NET-Version eines Windows-Sicherheitskontexts ist auf zwei Klassen aufgeteilt:

- **WindowsPrincipal –** In dieser Klasse sind die dem aktuellen Windows-Benutzer zugeordneten Rollen gespeichert. Die **WindowsPrincipal**-Implementierung behandelt Windows-Gruppen als Rollen. Die **IPrncipal.IsInRole**-Methode gibt basierend auf der Windows-Gruppenmitgliedschaft des Benutzers **True** oder **False** zurück.
- **WindowsIdentity –** Diese Klasse enthält den Identitätsteil des Sicherheitskontexts des aktuellen Benutzers und kann mit der statischen **WindowsIdentity.GetCurrent()**-Methode ermittelt werden. Diese Methode gibt ein **WindowsIdentity**-Objekt mit einer **Token**-Eigenschaft zurück, die einen **IntPtr** zurückgibt, der einen Windows-Handle auf das Zugriffstoken zurückgibt, das dem aktuellen Ausführungsthread zugeordnet ist. Dieses Token kann dann an die systemeigenen Win32®-API-Funktionen (Application Programming Interface) wie **GetTokenInformation**, **SetTokenInformation**, **CheckTokenMembership** usw. übergeben werden, um die Sicherheitsinformationen über das Token abzurufen.

Hinweis: Die statische **WindowsIdentity.GetCurrent()**-Methode gibt die Identität des aktuell ausgeführten Threads zurück, der die Identität wechseln kann oder nicht. Diese Methode ist mit der **GetUserName**-API von Win32 vergleichbar.

GenericPrincipal und zugeordnete Identitätsobjekte

Diese Implementierungen sind sehr einfach und werden von Anwendungen verwendet, die keine Windows-Authentifizierung verwenden und keine komplexen Darstellungen eines Principals benötigen. Sie können im Code sehr leicht erstellt werden. Daher muss ein gewisses Vertrauen bestehen, wenn eine Anwendung mit **GenericPrincipal** arbeitet.

Wenn Sie die **IsInRole**-Methode für **GenericPrincipal** verwenden, um Entscheidungen zur Autorisierung zu fällen, müssen Sie der Anwendung vertrauen, von der **GenericPrincipal** gesendet wurde. Dies verhält sich bei **WindowsPrincipal**-Objekten anders, bei denen dem Betriebssystem vertraut werden muss, um ein gültiges **WindowsPrincipal**-Objekt mit einer authentifizierten Identität und gültigen Gruppen-/Rollennamen bereitzustellen.

Der **GenericPrincipal**-Klasse können die folgenden Typen von Identitätsobjekten zugeordnet werden:

- **FormsIdentity –** Diese Klasse stellt eine Identität dar, die mit der Formularauthentifizierung authentifiziert wurde. Sie enthält ein **FormsAuthenticationTicket**, das Informationen über die Authentifizierungssitzung des Benutzers enthält.
- **PassportIdentity –** Diese Klasse stellt eine Identität dar, die mit der Passport-Authentifizierung authentifiziert wurde und Passport-Profilinformationen enthält.
- **GenericIdentity –** Diese Klasse stellt einen logischen Benutzer dar, der nicht an eine bestimmte Betriebssystemtechnologie gebunden ist und normalerweise in Zusammenhang mit benutzerdefinierten Authentifizierungs- und Autorisierungsmechanismen verwendet wird.

ASP.NET und HttpContext.User

Thread.CurrentPrincipal wird im .NET-Code normalerweise überprüft, bevor Autorisierungsentscheidungen gefällt werden. ASP.NET stellt den Sicherheitskontext des authentifizierten Benutzers jedoch mithilfe von **HttpContext.User** bereit.

Diese Eigenschaft übernimmt eine **IPrincipal**-Schnittstelle und gibt ebenfalls eine **IPrincipal**-Schnittstelle zurück. Die Eigenschaft enthält einen authentifizierten Benutzer für die aktuelle Anforderung. ASP.NET ruft **HttpContext.User** bei Autorisierungsentscheidungen ab.

Wenn die Windows-Authentifizierung verwendet wird, erstellt das Windows-Authentifizierungsmodul automatisch ein **WindowsPrincipal**-Objekt, das in **HttpContext.User** gespeichert wird. Bei anderen Authentifizierungsmechanismen wie Formular- oder Passport-Authentifizierung müssen Sie ein **GenericPrincipal**-Objekt erstellen und in **HttpContext.User** speichern.

ASP.NET-Identitäten

Zu jedem Zeitpunkt während der Ausführung einer ASP.NET-Webanwendung können während einer einzigen Anforderung mehrere Identitäten vorhanden sein. Dabei handelt es sich um die folgenden Identitäten:

- **HttpContext.User** gibt ein **IPrincipal**-Objekt zurück, das Sicherheitsinformationen für die aktuelle Webanforderung enthält. Hierbei handelt es sich um den authentifizierten Webclient.
- **WindowsIdentity.GetCurrent()** gibt die Identität des Sicherheitskontexts des aktuell ausgeführten Win32-Threads zurück. Diese Identität ist standardmäßig ASPNET, das Standardkonto, mit dem ASP.NET-Webanwendungen ausgeführt werden. Wenn die Webanwendung jedoch für Identitätswechsel konfiguriert wurde, stellt die Identität den authentifizierten Benutzer (bei anonymer IIS-Authentifizierung IUSR_COMPUTER).
- **Thread.CurrentPrincipal** gibt den Principal des aktuell, oberhalb des Win32-Threads ausgeführten .NET-Threads zurück.

Weitere Informationen

- Eine detaillierte Analyse der ASP.NET-Identität für eine Kombination von Webanwendungskonfigurationen (sowohl mit als auch ohne Identitätswechsel) finden Sie unter "ASP.NET-Identitätsmatrix" im Abschnitt "Referenz" dieses Handbuchs.
- Weitere Informationen zum Erstellen einer eigenen **IPrincipal**-Implementierung finden Sie in Kapitel 8, "ASP.NET-Sicherheit", und unter "Vorgehensweise: Implementieren von IPrincipal" im Abschnitt "Referenz" dieses Handbuchs.

Remoting und Webdienste

In der aktuellen Version des .NET Framework verfügen Remoting und Webdienste über kein eigenes Sicherheitsmodell. Beide übernehmen das Sicherheitsfeature von IIS und ASP.NET.

Obwohl in die Remotingarchitektur keine Sicherheit integriert ist, wurde die Sicherheit beim Entwurf berücksichtigt. Es bleibt dem Entwickler und/oder Administrator überlassen, in Remotinganwendungen gewisse Sicherheitsstufen zu integrieren. Ob Principalobjekte über Remotinggrenzen hinweg übergeben werden, hängt vom Standort des Client und des Remoteobjekts ab, z. B.:

- **Remoting im gleichen Prozess –** Wenn Remoting zwischen Objekten in derselben Anwendungsdomäne oder in getrennten Anwendungsdomänen verwendet wird, kopiert die Remotinginfrastruktur einen Verweis auf das **IPrincipal**-Objekt, das dem Kontext des Aufrufers zugeordnet ist, in den Kontext des Empfängers.
- **Prozessübergreifendes Remoting –** In diesem Fall werden zwischen den Prozessen keine **IPrincipal**-Objekte übertragen. Die Anmeldeinformationen, mit denen der ursprüngliche **IPrincipal** erstellt wurde, müssen an den Remoteprozess, der sich auf einem anderen Computer befinden kann, übertragen werden. Dadurch kann der Remotecomputer basierend auf den übergebenen Anmeldeinformationen ein geeignetes **IPrincipal**-Objekt erstellen.

Zusammenfassung

In diesem Kapitel wurden die Authentifizierungs- und Autorisierungsoptionen vollständig vorgestellt, die von den verschiedenen .NET-relevanten Technologien bereitgestellt werden. Durch die Verwendung von mehreren Gatekeepern in der .NET-Webanwendung können Sie eine Sicherheitsstrategie mit tief greifender Verteidigung implementieren. Zusammenfassend kann festgestellt werden:

- ASP.NET-Anwendungen können die vorhandenen Sicherheitsfeatures verwenden, die von Windows und IIS bereitgestellt werden.
- Mithilfe einer Kombination von SSL und IPSec kann eine sichere Kommunikation durch die Schichten einer .NET-Webanwendung ermöglicht werden, z. B. vom Browser zur Datenbank.
- Mit SSL können die unverschlüsselten Anmeldeinformationen geschützt werden, die über das Netzwerk übertragen werden, wenn die Standard- oder die Formularauthentifizierung verwendet wird.
- .NET stellt Benutzer, die mit der Windows-Authentifizierung authentifiziert wurden, mit einer Kombination der Klassen **WindowsPrincipal** und **WindowsIdentity** dar.
- Mit den Klassen **GenericPrincipal** und **GenericIdentity** oder **FormsIdentity** werden Benutzer dargestellt, die mit anderen Schemas als der Windows-Authentifizierung authentifiziert wurden, z. B. der Formularauthentifizierung.
- Sie können eigene Principal- und Identitätsimplementierungen erstellen, indem Sie Klassen erstellen, die **IPrincipal** und **IIdentity** implementieren.
- In ASP.NET-Webanwendungen wird das **IPrincipal**-Objekt, das den authentifizierten Benutzer darstellt, mit der **HttpContext.User**-Eigenschaft der aktuellen HTTP-Webanforderung zugeordnet.
- Bei Gates handelt es sich um Zugriffssteuerungspunkte in der Anwendung, durch die autorisierte Benutzer auf Ressourcen oder Dienst zugreifen können. Gatekeeper sind für die Steuerung des Zugriffs auf Gates verantwortlich.
- Mit mehreren Gatekeepern kann eine tief greifende Verteidigungsstrategie bereitgestellt werden.

Das nun folgende Kapitel 3, "Authentifizierung und Autorisierung", enthält zusätzliche Informationen, die Ihnen bei der Auswahl der geeignetsten Authentifizierungs- und Autorisierungsstrategie für das jeweilige Anwendungsszenario helfen.