Erstellen sicherer ASP.NET-Anwendungen

Authentifizierung, Autorisierung und sichere Kommunikation

Kapitel 7 - Internetsicherheit

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

Microsoft Corporation

Oktober 2002

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

Zusammenfassung

In diesem Kapitel wird beschrieben, wie häufige Internetanwendungsszenarien gesichert werden. Es enthält eine Darstellung der Merkmale der einzelnen Szenarien und eine Beschreibung der erforderlichen Schritte, um das Szenario zu sichern. Weitere Informationen finden Sie in den entsprechenden Analyseabschnitten.

Inhalt

Internetanwendungen zeichnen sich durch eine große Zielgruppe, viele potenzielle Benutzer und unterschiedliche Sicherheitsanforderungen aus. Sie reichen von Portalanwendungen, die keine Benutzerauthentifizierung erfordern, über Webanwendungen, die registrierten Benutzern Inhalte bereitstellen, bis hin zu weiträumigen E-Commerce-Anwendungen, die umfassende Authentifizierung, Autorisierung, Kreditkartenüberprüfung und sichere Übertragung vertraulicher Daten über öffentliche und interne Netzwerke erfordern.

Als Entwickler von Internetanwendungen müssen Sie sich der Herausforderung stellen, sicherzustellen, dass die Anwendung geeignete Verteidigungsmechanismen verwendet und skalierbar, hochleistungsfähig und sicher ist. Zu diesen Herausforderungen zählen:

- Auswählen eines geeigneten Speichers für die Anmeldeinformationen der Benutzer, z. B. eine benutzerdefinierte Datenbank oder der Verzeichnisdienst Active Directory®.
- Sicherstellen der Funktionsfähigkeit der Anwendung durch Firewalls.
- Übertmitteln der Anmeldeinformationen für die Sicherheit durch die Ebenen der Anwendung.
- Durchführen der Autorisierung.
- Sicherheit der Integrität der Daten und des Datenschutzes bei der Übertragung über öffentliche und interne Netzwerke.
- Sichern des Anwendungsstatus mit einer Datenbank.
- Sicherstellen der Integrität der Anwendungsdaten.
- Implementieren einer Lösung, die an eine möglicherweise hohe Anzahl von Benutzern angepasst werden kann.

Die beiden in diesem Kapitel behandelten, gebräuchlichen Internetanwendungsszenarien, anhand derer die empfohlenen Techniken für Authentifizierung, Autorisierung und sichere Kommunikation veranschaulicht werden, sind:

- ASP.NET zu SQL Server
- ASP.NET über Remote Enterprise Services zu SQL Server

ASP.NET zu SQL Server

In diesem Szenario mit zwei physischen Ebenen melden sich registrierte Benutzer mit einem Webbrowser sicher bei der webbasierten Anwendung an. Die ASP.NET-basierte Webanwendung verwendet sichere Verbindungen zu einer Microsoft® SQL Server™-Datenbank, um überwiegend den Abruf von Daten zu verwalten. Ein Beispiel dafür ist eine Portalanwendung, die registrierten Abonnenten Newsinhalte bereitstellt (siehe Abbildung 7.1).

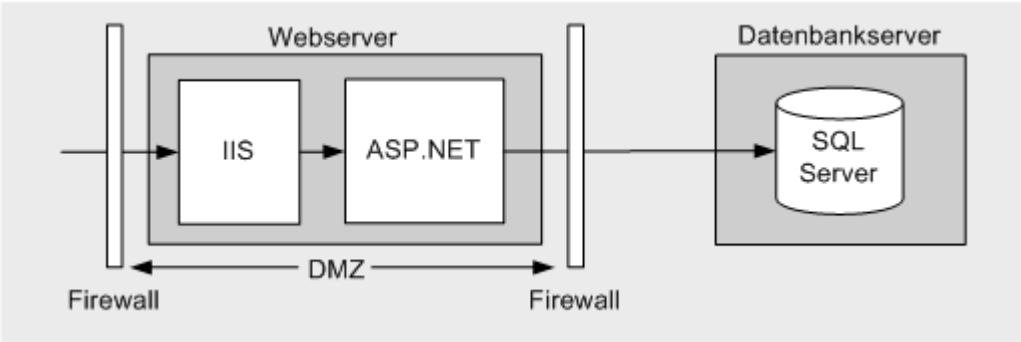

Abbildung 7.1
Internetszenario: ASP.NET-Webanwendung zu SQL Server

Merkmale

Dieses Szenario weist die folgenden Merkmale auf:

- Die Benutzer verwenden eine Vielzahl verschiedener Browsertypen.
- Anonyme Benutzer können die uneingeschränkten Seiten der Anwendung durchsuchen.
- Die Benutzer müssen sich registrieren oder anmelden (über ein HTML-Formular), bevor sie eingeschränkte Seiten anzeigen dürfen.
- Die Anmeldeinformationen der Benutzer werden anhand einer SQL Server-Datenbank überprüft.
- Alle Benutzereingaben (z. B. die Anmeldeinformationen der Benutzer), die in Datenbankabfragen verwendet werden, werden überprüft, um die Gefahr von SQL Injection-Angriffen zu mindern.

- Die Front-End-Webanwendung befindet sich in einem Perimeternetzwerk (auch DMZ, demilitarisierte Zone und abgeschirmtes Subnetz genannt), wobei die Trennung vom Internet und dem internen Firmennetzwerk (sowie der SQL Server-Datenbank) durch Firewalls erfolgt.
- Die Anwendung erfordert strenge Sicherheit, eine hohe Skalierbarkeit und eine detaillierte Überwachung.
- Die Datenbank vertraut der Anwendung, dass die Benutzer ordnungsgemäß authentifiziert werden (d. h. die Anwendung führt die Datenbankaufrufe im Namen der Benutzer durch).
- Die Webanwendung stellt die Verbindungen zu der Datenbank unter Verwendung des ASP.NET-Prozesskontos her.
- In SQL Server wird für die Datenbankautorisierung eine einzige benutzerdefinierte Datenbankrolle verwendet.

Sichern des Szenarios

In diesem Szenario stellt die Webanwendung eine Anmeldeseite dar, um die Anmeldeinformationen zu übernehmen. Erfolgreich überprüfte Benutzer dürfen fortfahren. Allen anderen wird der Zugriff verweigert. Die Datenbank führt die Authentifizierung anhand der ASP.NET-Standardprozessidentität durch. Dabei handelt es sich um ein Konto mit wenigen Rechten (d. h. die Datenbank vertraut der ASP.NET-Anwendung).

Tabelle 7.1: *Übersicht über die Sicherheit*

Kategorie	Details
Authentifizierung	• IIS ist so konfiguriert, dass ein anonymer Zugriff möglich ist. Die ASP.NET-Webanwendung authentifiziert die Benutzer mit der Formularauthentifizierung für die Eingabe der Anmeldeinformationen. Die Überprüfung erfolgt anhand einer SQL Server-Datenbank. • Die Kennwörter der Benutzer sind nicht in der Datenbank gespeichert. Stattdessen werden Kennworthashes mit Salt-Werten gespeichert. Der Salt-Wert mindert die Gefahr von Verzeichnisangriffen (Wörterbuchangriffen). • Für Verbindungen zur Datenbank wird die Windows®-Authentifizierung unter Verwendung des Windows-Kontos mit geringst möglichen Rechten verwendet, mit dem die ASP.NET-Webanwendung ausgeführt wird.
Autorisierung	• Das ASP.NET-Prozesskonto ist autorisiert, auf die Systemressourcen des Webservers zuzugreifen. Die Ressourcen sind durch Windows-ACLs geschützt. • Der Zugriff auf die Datenbank wird durch die Identität der ASP.NET-Anwendung autorisiert.
Sichere Kommunikation	• Die vertraulichen Daten, die zwischen den Benutzern und der Webanwendung gesendet werden, werden mit SSL gesichert. • Die vertraulichen Daten, die zwischen dem Webserver und dem Datenbankserver gesendet werden, werden mit IPSec gesichert.

Das Ergebnis

Abbildung 7.2 zeigt die empfohlene Sicherheitskonfiguration für dieses Szenario.

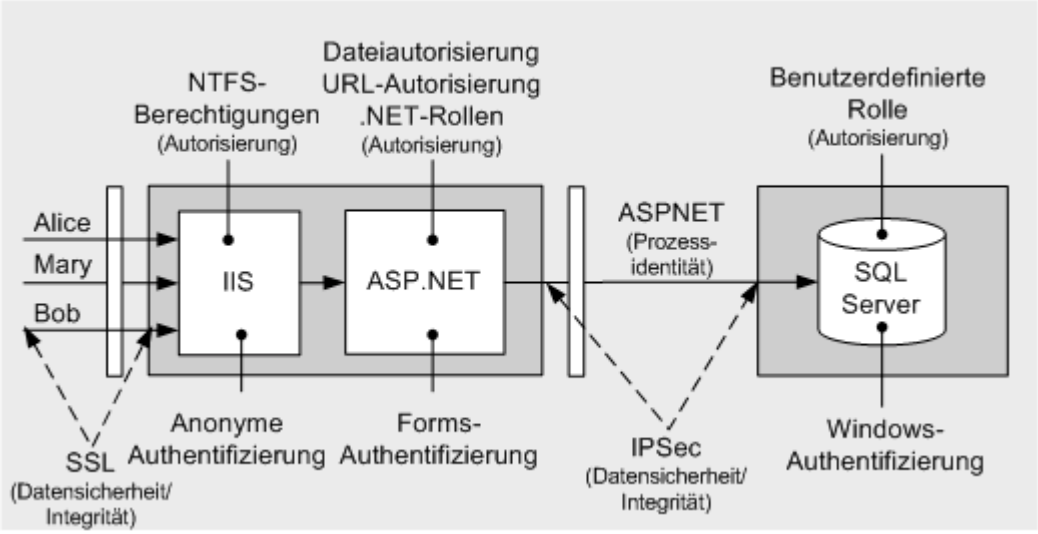

Abbildung 7.2

Internetszenario: ASP.NET zu SQL Server - Empfohlene Sicherheitskonfiguration

Konfigurationsschritte für die Sicherheit

Bevor Sie beginnen, sollten Sie sich über folgende Themen informieren:

- Erstellen benutzerdefinierter ASP.NET-Konten (siehe "Vorgehensweise: Erstellen eines benutzerdefinierten Kontos zum Ausführen von ASP.NET" im Abschnitt "Referenz" dieses Handbuchs)
- Erstellen eines Datenbankkontos mit möglichst geringen Rechten (siehe Kapitel 12, "Datenzugriffssicherheit")
- Konfigurieren von SSL auf einem Webserver (siehe "Vorgehensweise: Einrichten von SSL auf einem Webserver" im Abschnitt "Referenz" dieses Handbuchs)
- Konfigurieren von IPSec (siehe "Vorgehensweise: Verwenden von IPSec zum Sichern der Kommunikation zwischen zwei Servern" im Abschnitt "Referenz" dieses Handbuchs)

Konfigurieren des Webservers

Konfigurieren von IIS	
Schritt	Weitere Informationen
Aktivieren des anonymen Zugriffs für das virtuelle Stammverzeichnis der Webanwendung	Die IIS-Authentifizierungseinstellungen können Sie mit dem IIS-MMC-Snap-In bearbeiten. Klicken Sie mit der rechten Maustaste auf das virtuelle Verzeichnis der Anwendung, und klicken Sie dann auf **Eigenschaften**. Klicken Sie auf die Registerkarte **Verzeichnissicherheit** und dann im Gruppenfeld **Authentifizierung und Zugriffssteuerung** auf **Bearbeiten**.

Konfigurieren von ASP.NET	
Schritt	Weitere Informationen

Zurücksetzen des Kennworts des Kontos **ASPNET** (unter dem ASP.NET ausgeführt wird) auf ein bekanntes starkes Kennwort	Dadurch können Sie ein doppeltes lokales Konto (mit demselben Benutzernamen und demselben Kennwort) auf dem Datenbankserver erstellen. Dies ist erforderlich, damit das Konto **ASPNET** Authentifizierungsanforderungen des Datenbankservers über das Netzwerk beantworten kann, wenn die Verbindung mit der Windows-Authentifizierung hergestellt wird.
	
	Als Alternative kann hier ein Domänenkonto mit möglichst geringen Rechten verwendet werden (falls eine Windows-Authentifizierung durch den Firewall zulässig ist).
	Weitere Informationen finden Sie unter "Prozessidentität für ASP.NET" in Kapitel 8, "ASP.NET-Sicherheit".
	
	Bearbeiten Sie **Machine.config** in **%windir%\Microsoft.NET\Framework\v1.0.3705\CONFIG**.
	
	Legen Sie die Attribute für den Benutzernamen und das Kennwort des benutzerdefinierten Kontos im <**processModel**>-Element fest.
	
	Die Standardeinstellung
	
	`<!-- userName="machine" password="AutoGenerate" -->`
	
	wird geändert in
	
	`<!-- userName="machine"`
	`password="YourStrongPassword" -->`

Konfigurieren der ASP.NET-Webanwendung, sodass die Formularauthentifizierung (mit SSL) verwendet wird	Bearbeiten Sie **Web.config** im virtuellen Stammverzeichnis der Anwendung.
	Legen Sie das <**authentication**>-Element fest auf:
	
	`<authentication mode="Forms" >`
	`<forms name="MyAppFormsAuth"`
	`loginUrl="login.aspx"`
	`protection="All"`
	`timeout="20"`
	`path="/" >`
	`</forms>`
	`</authentication>`
	
	Weitere Informationen zum Verwenden der Formularauthentifizierung anhand einer SQL Server-Datenbank finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit SQL Server 2000" im Abschnitt "Referenz" dieses Handbuchs.

Konfigurieren von SQL Server

Schritt	Weitere Informationen
Erstellen eines Windows-Kontos auf dem SQL Server-Computer, das mit dem ASP.NET-Prozesskonto übereinstimmt	Benutzername und Kennwort müssen mit dem benutzerdefinierten ASP.NET-Anwendungskonto bzw. mit **ASPNET** übereinstimmen, falls das Standardkonto verwendet wird.
Konfigurieren von SQL Server für die Windows-Authentifizierung	
Erstellen eines SQL Server-Benutzernamens für das benutzerdefinierte ASP.NET-Anwendungskonto	Dadurch wird der Zugriff auf SQL Server gewährt.
Erstellen eines neuen Datenbankbenutzers und Zuordnen des Benutzernamens zum Datenbankbenutzer	Dadurch wird der Zugriff auf die angegebene Datenbank gewährt.
Erstellen einer neuen benutzerdefinierten Datenbankrolle in der Datenbank und Einfügen des Datenbankbenutzers in die Rolle	
Einrichten der Datenbankberechtigungen für die Datenbankrolle	Gewähren Sie minimale Rechte. Weitere Informationen finden Sie in Kapitel 12, "Datenzugriffssicherheit".

Konfigurieren der sicheren Kommunikation

Schritt	Weitere Informationen
Konfigurieren der Website für SSL	Siehe "Vorgehensweise: Einrichten von SSL auf einem Webserver" im Abschnitt "Referenz" dieses Handbuchs.
Konfigurieren von IPSec zwischen Anwendungsserver und Datenbankserver	Siehe "Vorgehensweise: Verwenden von IPSec zum Sichern der Kommunikation zwischen zwei Servern" im Abschnitt "Referenz" dieses Handbuchs.

Analyse

- Die Formularauthentifizierung ist in diesem Szenario ideal, da die Benutzer über keine Windows-Konten verfügen. Auf der Formularanmeldeseite werden die Anmeldeinformationen der Benutzer eingegeben. Die Überprüfung der Anmeldeinformationen muss durch den Anwendungscode durchgeführt werden. Dabei kann ein beliebiger Datenspeicher verwendet werden. Meist wird eine SQL Server-Datenbank verwendet, obwohl Active Directory eine Alternative für die Speicherung der Anmeldeinformationen bereitstellt.

- Bei der Formularauthentifizierung müssen Sie die ersten Anmeldeinformationen mit SSL schützen. Das Formularauthentifizierungsticket (das bei nachfolgenden Webanforderungen vom authentifizierten Client als Cookie übertragen wird) muss ebenfalls geschützt werden. Sie können SSL für alle Seiten verwenden, um das Ticket zu schützen. Alternativ können Sie das Formularauthentifizierungsticket verschlüsseln, indem Sie das **protection**-Attribut des <**forms**>-Elements (**All** oder **Encrypt**) konfigurieren und das Ticket mit der **Encrypt**-Methode der **FormsAuthentication**-Klasse verschlüsseln.

  Das **protection="All"**-Attribut gibt an, dass das Ticket überprüft (Integritätsprüfung) und verschlüsselt werden soll, wenn die Anwendung **FormsAuthentication.Encrypt** aufruft. Rufen Sie diese Methode auf, wenn Sie das Authentifizierungsticket erstellen, normalerweise im Ereignishandler für die Anmeldeschaltfläche der Anwendung.

  ```
  string encryptedTicket = FormsAuthentication.Encrypt(authTicket);
  ```

  Weitere Informationen zur Formularauthentifizierung und Ticketverschlüsselung finden Sie in Kapitel 8, "ASP.NET-Sicherheit".
- ASP.NET wird als lokales Konto **ASPNET** mit möglichst geringen Rechten ausgeführt, sodass ein mögliches Risiko einer Offenlegung gemindert wird.
- Die URL-Autorisierung im Webserver ermöglicht nicht authentifizierten Benutzern, unbeschränkte Webseiten zu durchsuchen, und erzwingt eine Authentifizierung für beschränkte Seiten.
- Da keine Identitätswechsel aktiviert sind, erfolgt jeder Zugriff auf lokale oder Remoteressourcen durch die webbasierte Anwendung im Sicherheitskontext des Kontos **ASPNET**. Die Windows-ACLs für sichere Ressourcen müssen entsprechend eingerichtet werden.
- Die Anmeldeinformationen der Benutzer werden anhand einer benutzerdefinierten SQL Server-Datenbank überprüft. In der Datenbank werden Kennworthashes (mit Salt-Wert) gespeichert. Weitere Informationen finden Sie unter "Authentifizieren von Benutzern anhand einer Datenbank" in Kapitel 12, "Datenzugriffssicherheit".
- Durch die Verwendung der Windows-Authentifizierung bei SQL Server vermeiden Sie das Speichern der Anmeldeinformationen in Dateien auf dem Webserver und ebenfalls deren Übertragung über das Netzwerk.
- Wenn die Anwendung zurzeit die SQL-Authentifizierung verwendet, müssen Sie die Datenbankverbindungszeichenfolge sicher speichern, da sie Benutzernamen und Kennwörter enthält. Sie sollten erwägen, DPAPI zu verwenden. Weitere Informationen finden Sie unter "Storing Database Connection Strings Securely" in Kapitel 12, "Datenzugriffssicherheit".
- Die Verwendung eines duplizierten Windows-Kontos im Datenbankserver (das mit dem ASP.NET-Prozesskonto übereinstimmt) führt zu einem gesteigerten Verwaltungsaufwand. Wenn das Kennwort auf einem Computer geändert wird, muss es synchronisiert und auf allen Computern aktualisiert werden. In einigen Szenarien können Sie eventuell ein Domänenkonto mit möglichst geringen Rechten verwenden, um die Verwaltung zu vereinfachen.
- IPSec zwischen Webserver und Datenbankserver stellt den Schutz der Daten sicher, die von und zu der Datenbank gesendet werden.
- SSL zwischen Browser und Webserver schützt die Anmeldeinformationen und andere sicherheitssensitive Daten wie Kreditkartennummern.
- Stellen Sie sicher, dass die Verschlüsselungsschlüssel, z. B. diejenigen, mit denen das Formularauthentifizierungsticket verschlüsselt wird (und durch das <**machineKey**>-Element in **Machine.config** angegeben werden), auf allen Servern in der Farm übereinstimmen, wenn Sie eine Webfarm verwenden. Weitere Details zur Verwendung von ASP.NET in einem Webfarmszenario finden Sie in Kapitel 8, "ASP.NET-Sicherheit".

Fallstricke

Die Anwendung muss die Identität des ursprünglichen Aufrufers an die Datenbank übertragen, damit die Anforderungen der Überwachung unterstützt werden. Die Identität des Aufrufers kann mithilfe von Parametern gespeicherter Prozeduren übergeben werden.

Verwandte Szenarien

Formularauthentifizierung anhand Active Directory

Die Anmeldeinformationen der Benutzer, die von der Formularanmeldeseite übernommen werden, können anhand einer Vielzahl von Speichern authentifiziert werden. Active Directory bildet eine Alternative zur Verwendung einer SQL Server-Datenbank.

Weitere Informationen

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

.NET-Rollen für die Autorisierung

In dem obigen Szenario werden nicht die verschiedenen Benutzertypen berücksichtigt, die auf die Anwendung zugreifen. Ein Portalserver kann beispielsweise verschiedene Abonnementsebenen aufweisen wie Standard, Premier oder Enterprise.

Wenn die Rolleninformationen im Benutzerspeicher (SQL Server-Datenbank) verwaltet werden, kann die Anwendung ein **GenericPrincipal**-Objekt erstellen, in dem Rollen- und Identitätsinformationen gespeichert werden können. Nachdem **GenericPrincipal** erstellt und in den Kontext der Webanforderung (mit **HttpContext.User**) eingefügt wurde, können Sie programmatische Rollenprüfungen zum Methodencode hinzufügen oder Methoden und Seiten mit **PrincipalPermission**-Attributen erweitern, um eine bestimmte Rollenmitgliedschaft zu fordern.

Weitere Informationen

- Weitere Informationen zum Erstellen von **GenericPrincipal**-Objekten, die Rollenlisten enthalten, finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit GenericPrincipal-Objekten" im Abschnitt "Referenz" dieses Handbuchs.
- Weitere Informationen zu **PrincipalPermission**-Forderungen und programmatischen Rollenprüfungen finden Sie in Kapitel 8, "ASP.NET-Sicherheit".

Verwenden eines anonymen Domänenkontos auf dem Webserver

In diesem geänderten Szenario wird das standardmäßige anonyme Internetbenutzerkonto (ein lokales Konto mit dem Namen **IUSR_COMPUTER**) durch ein Domänenkonto ersetzt. Das Domänenkonto wird mit den minimal erforderlichen Rechten konfiguriert, damit die Anwendung ausgeführt werden kann (Sie können ohne Rechte beginnen und schrittweise Rechte hinzufügen). Wenn mehrere webbasierte Anwendungen vorhanden sind, können Sie andere Domänenkonten verwenden (jeweils eins für jedes webbasierte Anwendung oder jedes virtuelle Verzeichnis).

Aktivieren Sie mithilfe der folgenden Einstellung der Datei **web.config** Identitätswechsel für die webbasierte Anwendung, damit der Sicherheitskontext des anonymen Domänenkontos von IIS an ASP.NET gelangt:

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

Wenn die webbasierte Anwendung mit einer Remoteressource kommuniziert, z. B. einer Datenbank, müssen dem Domänenkonto die erforderlichen Berechtigungen für die Ressource erteilt werden. Wenn die Anwendung beispielsweise auf ein Remotedateisystem zugreift, müssen die ACLs entsprechend konfiguriert werden, sodass das Domänenkonto (zumindest) Lesezugriff erhält. Wenn die Anwendung auf eine SQL Server-Datenbank zugreift, muss das Domänenkonto mit einem SQL-Benutzernamen einem Datenbankanmeldenamen zugeordnet werden.

Da der Sicherheitskontext, der die Anwendung durchläuft, vom anonymen Konto stammt, muss die Identität des ursprünglichen Aufrufers (die durch die Formularauthentifizierung erfasst wurde) auf Anwendungsebene von Ebene zu Ebene übertragen werden, beispielsweise über Parameter von Methoden und gespeicherten Prozeduren.

Weitere Informationen

- Weitere Informationen zu diesem Verfahren finden Sie unter "Verwenden des anonymen Internetbenutzerkontos" in Kapitel 8, "ASP.NET-Sicherheit".
- Bevor Sie dieses Szenario implementieren, sollten Sie den Artikel Q259353 "Must Enter Password Manually After You Set Password Synchronization" (US) in der Microsoft Knowledge Base lesen.

ASP.NET über Remote Enterprise Services zu SQL Server

In diesem Szenario stellt ein Webserver, auf dem ASP.NET-Seiten ausgeführt werden, sichere Verbindungen zu Serviced Components her, die sich auf einem Remoteanwendungsserver befinden, der wiederum eine Verbindung zu einer SQL Server-Datenbank herstellt. Wie bei vielen Internetanwendungsinfrastrukturen sind Webserver und Anwendungsserver durch einen Firewall getrennt, und der Webserver befindet sich in einem Perimeternetzwerk. Die Serviced Components stellen sichere Verbindungen zu SQL Server her.

Betrachten Sie als Beispiel eine Internetbankanwendung, die Benutzern vertrauliche Daten (z. B. Details zu privaten Finanzen) bereitstellt. Alle Banktransaktionen vom Client zur Datenbank müssen gesichert werden, und die Datenintegrität ist von entscheidender Bedeutung. Nicht nur der Datenverkehr vom und zum Benutzer muss gesichert werden, sondern auch der Datenverkehr von und zu der Datenbank (siehe Abbildung 7.3).

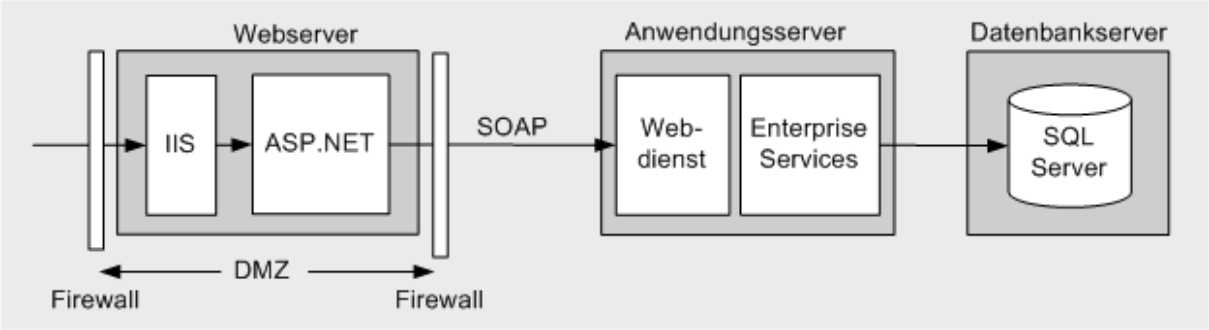

Abbildung 7.3
Internetszenario: ASP.NET über Remote Enterprise Services zu SQL Server

Merkmale

- Die Benutzer verwenden eine Vielzahl verschiedener Browsertypen.
- Anonyme Benutzer können die uneingeschränkten Seiten der Anwendung durchsuchen.
- Die Benutzer müssen sich registrieren oder anmelden (über ein HTML-Formular), bevor sie eingeschränkte Seiten anzeigen dürfen.

- Die Front-End-Webanwendung befindet sich in einem Perimeternetzwerk, wobei die Trennung vom Internet und dem internen Firmennetzwerk (sowie dem Anwendungsserver) durch Firewalls erfolgt.
- Die Anwendung erfordert strenge Sicherheit, eine hohe Skalierbarkeit und eine detaillierte Überwachung.
- Die webbasierte Anwendung verwendet SOAP für die Verbindung zur Webdienstschicht, die eine Schnittstelle zu den Serviced Components bereitstellt, die in einer Enterprise Services-Anwendung auf dem Anwendungsserver ausgeführt werden. SOAP wird aufgrund von Firewalleinschränkungen gegenüber DCOM bevorzugt.
- SQL Server verwendet für die Autorisierung eine einzige benutzerdefinierte Datenbankrolle.
- Die Daten sind sicherheitssensitiv. Integrität und Datenschutz muss über das Netzwerk und in allen persistenten Datenspeichern gewährleistet werden.
- Die Datenintegrität wird mithilfe von Enterprise Services (COM+)-Transaktionen erzwungen.

Sichern des Szenarios

In diesem Szenario übernimmt der Webdienst die Anmeldeinformationen von einer Formularanmeldeseite. Der Aufrufer wird anschließend anhand einer SQL Server-Datenbank authentifiziert. Die Anmeldeseite verwendet SSL, um die über das Internet übertragenen Anmeldeinformationen der Benutzer zu schützen.

Die webbasierte Anwendung kommuniziert mit einem Webdienst, der eine Schnittstelle zu den Unternehmensdiensten bereitstellt, die in Serviced Components implementiert sind. Der Webdienst vertraut der webbasierten Anwendung (im Perimeternetzwerk) und authentifiziert die ASP.NET-Prozessidentität. Die Identität des Benutzers durchläuft alle Ebenen auf der Anwendungsebene, wobei Parameter von Methoden und gespeicherten Prozeduren verwendet werden. Diese Informationen werden zur Überwachung der Aktionen des Benutzers in allen Ebenen verwendet.

Tabelle 7.2: *Sicherheitsmaßnahmen*

Kategorie	Details
Authentifizierung	• Stellen Sie auf dem Webserver eine starke Authentifizierung bereit. • Authentifizieren Sie die Identität der Enterprise Services-Anwendung an der Datenbank. • IIS ist für einen anonymen Zugriff konfiguriert. Die webbasierte Anwendung authentifiziert die Benutzer mit der Formularauthentifizierung (anhand einer SQL Server-Datenbank). • Das virtuelle Verzeichnis des Webdienstes ist für die integrierte Windows-Authentifizierung konfiguriert. Die Webdienste authentifizieren die Prozessidentität der webbasierten Anwendung. • Für die Verbindung zur Datenbank wird die Windows-Authentifizierung verwendet. Die Datenbank authentifiziert das Windows-Konto mit möglichst geringen Rechten, mit dem die Enterprise Services-Anwendung ausgeführt wird.

Autorisierung	• Es wird das Modell mit vertrauenswürdigen Subsystemen verwendet. Die Autorisierung der einzelnen Benutzer erfolgt nur in der Webanwendung. • Der Benutzerzugriff auf die Seiten im Webserver wird durch die URL-Autorisierung gesteuert. • Das ASP.NET-Prozesskonto ist autorisiert, auf die Systemressourcen des Webservers zuzugreifen. Die Ressourcen sind durch ACLs geschützt. • Die Berechtigungen in der Datenbank werden durch eine benutzerdefinierte Rolle gesteuert. Die Identität der Enterprise Services-Anwendung ist Mitglied dieser Rolle. • Das Enterprise Services-Prozesskonto ist autorisiert, auf die Systemressourcen des Anwendungsservers zuzugreifen. Die Ressourcen sind durch ACLs geschützt.
Sichere Kommunikation	• Die vertraulichen Daten, die zwischen den Benutzern und der webbasierten Anwendung gesendet werden, werden mit SSL gesichert. • Die vertraulichen Daten, die zwischen dem Webserver und dem Webdienst gesendet werden, werden mit SSL gesichert. • Die vertraulichen Daten, die zwischen den Serviced Components und der Datenbank gesendet werden, werden mit IPSec gesichert.

Das Ergebnis

Abbildung 7.4 zeigt die empfohlene Sicherheitskonfiguration für dieses Szenario.

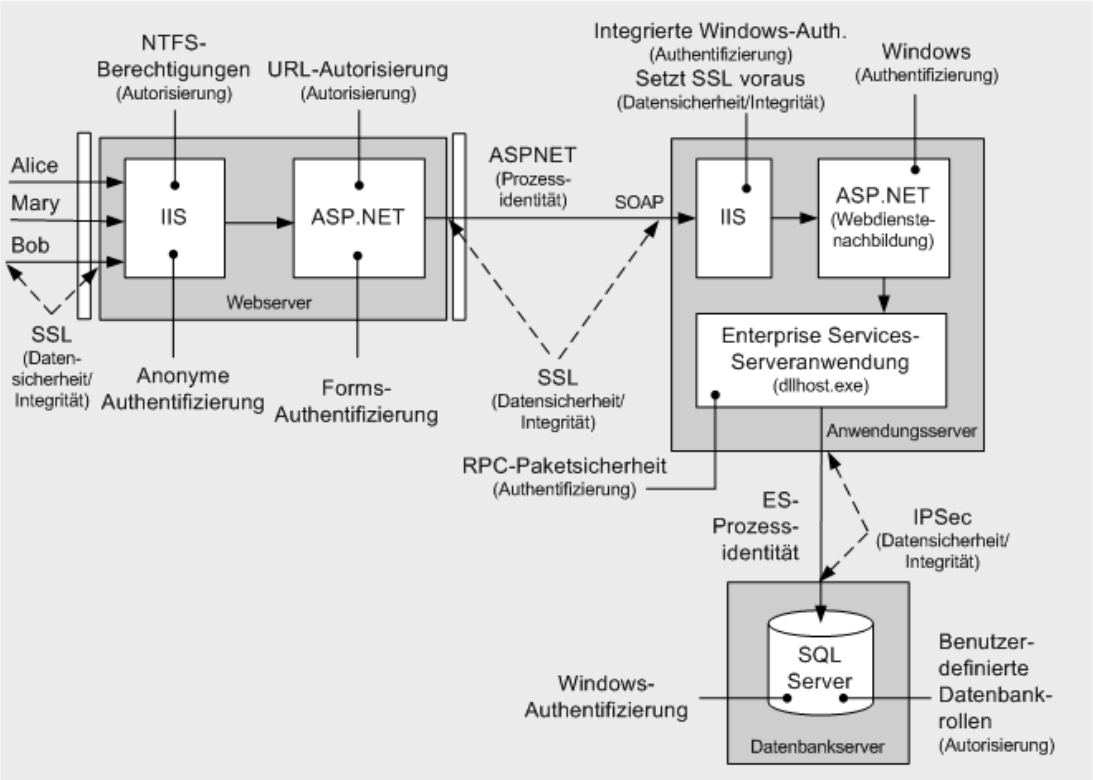

Abbildung 7.4

Internetszenario: ASP.NET über Remote Enterprise Services zu SQL Server - Empfohlene Sicherheitskonfiguration

Konfigurationsschritte für die Sicherheit

Bevor Sie beginnen, sollten Sie sich über folgende Themen informieren:

- Erstellen eines Datenbankkontos mit möglichst geringen Rechten (siehe Kapitel 12, "Datenzugriffssicherheit")
- Konfigurieren von SSL auf einem Webserver (siehe "Vorgehensweise: Einrichten von SSL auf einem Webserver" im Abschnitt "Referenz" dieses Handbuchs)
- Konfigurieren von IPSec (siehe "Vorgehensweise: Verwenden von IPSec zum Sichern der Kommunikation zwischen zwei Servern" im Abschnitt "Referenz" dieses Handbuchs)
- Konfigurieren der Enterprise Services-Sicherheit (siehe "Vorgehensweise: Verwenden von rollenbasierter Sicherheit mit Enterprise Services" im Abschnitt "Referenz" dieses Handbuchs)

Konfigurieren des Webservers

Konfigurieren von IIS	
Schritt	**Weitere Informationen**
Aktivieren des anonymen Zugriffs für das virtuelle Stammverzeichnis der webbasierten Anwendung	
Konfigurieren von ASP.NET	
Schritt	**Weitere Informationen**

Zurücksetzen des Kennworts des Kontos **ASPNET** (unter dem ASP.NET ausgeführt wird) auf ein bekanntes starkes Kennwort	Dadurch können Sie ein doppeltes lokales Konto (mit demselben Benutzernamen und demselben Kennwort) auf dem Anwendungsserver erstellen. Dies ist erforderlich, damit das Konto **ASPNET** Authentifizierungsanforderungen des Anwendungsservers über das Netzwerk beantworten kann.
	
	Als Alternative kann ein Domänenkonto mit möglichst geringen Rechten verwendet werden (falls eine Windows-Authentifizierung durch den Firewall zulässig ist).
	
	Weitere Informationen finden Sie unter "Prozessidentität für ASP.NET" in Kapitel 8, "ASP.NET-Sicherheit".
	
	Bearbeiten Sie **Machine.config** in **%windir%\Microsoft.NET\Framework\v1.0.3705\CONFIG**.
	
	Legen Sie die Attribute für den Benutzernamen und das Kennwort des benutzerdefinierten Kontos im <**processModel**>-Element fest.
	
	Die Standardeinstellung
	
	`<!-- userName="machine" password="AutoGenerate" -->`
	
	wird geändert in
	
	`<!-- userName="machine"`
	`password="YourStrongPassword" -->`

Konfigurieren der webbasierten Webanwendung, sodass die Formularauthentifizierung (mit SSL) verwendet wird	Bearbeiten Sie **Web.config** im virtuellen Stammverzeichnis der Anwendung.
	Legen Sie das <**authentication**>-Element fest auf:
	
	`<authentication mode="Forms" >`
	`  <forms name="MyAppFormsAuth"`
	`  loginUrl="login.aspx"`
	`  protection="All"`
	`  timeout="20"`
	`  path="/" >`
	`  </forms>`
	`</authentication>`
	
	Weitere Informationen zum Verwenden der Formularauthentifizierung anhand einer SQL Server-Datenbank finden Sie unter "Vorgehensweise: Verwenden der Formularauthentifizierung mit SQL Server 2000" im Abschnitt "Referenz" dieses Handbuchs.

Konfigurieren des Anwendungsservers

Konfigurieren von IIS	
Schritt	Weitere Informationen
Deaktivieren des anonymen Zugriffs	
Konfigurieren der integrierten Windows-Authentifizierung	IIS authentifiziert die ASP.NET-Prozessidentität der webbasierten Anwendung auf dem Webserver.

Konfigurieren von ASP.NET	
Schritt	Weitere Informationen
Verwenden der Windows-Authentifizierung	Bearbeiten Sie **Web.config** im virtuellen Stammverzeichnis des Webdienstes. Legen Sie das <**authentication**>-Element fest auf: `<authentication mode="Windows" />`

Konfigurieren der Enterprise Services	
Schritt	Weitere Informationen
Erstellen eines benutzerdefinierten Kontos mit möglichst geringen Rechten für die Ausführung der Enterprise Services-Serveranwendung	**HINWEIS:** Wenn Sie ein lokales Konto verwenden, müssen Sie auf dem Datenbankservercomputer ebenfalls ein dupliziertes Konto erstellen.
Konfigurieren der Enterprise Services-Anwendung, sodass das benutzerdefinierte Konto verwendet wird	Siehe "Konfigurieren der Sicherheit" in Kapitel 9, "Enterprise Services-Sicherheit".
Aktivieren der rollenbasierten Zugriffsüberprüfung	Siehe "Konfigurieren der Sicherheit" in Kapitel 9, "Enterprise Services-Sicherheit".

Hinzufügen einer einzelnen Enterprise Services (COM+)-Rolle für die aufgerufene Anwendung (z. B. **Vertrauenswürdiger Webdienst**)	Die vollständige Autorisierung der Endbenutzer wird von der webbasierten Anwendung durchgeführt. Der Webdienst (und die Serviced Components) lassen Zugriff nur für Mitglieder der Rolle **Vertrauenswürdiger Webdienst** zu.
Hinzufügen des lokalen **ASPNET**-Kontos zu der Rolle **Vertrauenswürdiger Webdienst**	Siehe "Konfigurieren der Sicherheit" in Kapitel 9, "Enterprise Services-Sicherheit".

Konfigurieren von SQL Server

Schritt	Weitere Informationen
Erstellen eines Windows-Kontos auf dem SQL Server-Computer, das mit dem Enterprise Services-Anwendungskonto übereinstimmt	Benutzername und Kennwort müssen mit dem benutzerdefinierten Enterprise Services-Konto übereinstimmen.
Konfigurieren von SQL Server für die Windows-Authentifizierung	
Erstellen eines SQL Server-Benutzernamens für das benutzerdefinierte Enterprise Services-Konto	Dadurch wird der Zugriff auf SQL Server gewährt.
Erstellen eines neuen Datenbankbenutzers und Zuordnen des Benutzernamens zum Datenbankbenutzer	Dadurch wird der Zugriff auf die angegebene Datenbank gewährt.
Erstellen einer neuen benutzerdefinierten Datenbankrolle und Hinzufügen des Datenbankbenutzers zur Rolle	
Einrichten der Datenbankberechtigungen für die Datenbankrolle	Gewähren Sie minimale Rechte. Weitere Informationen finden Sie in Kapitel 12, "Datenzugriffssicherheit".

Konfigurieren der sicheren Kommunikation

Schritt	Weitere Informationen
Konfigurieren der Website für SSL	Siehe "Vorgehensweise: Einrichten von SSL auf einem Webserver" im Abschnitt "Referenz" dieses Handbuchs.
Konfigurieren von SSL zwischen Webserver und Anwendungsserver.	Siehe "Vorgehensweise: Aufrufen eines Webdienstes unter Verwendung von SSL" im Abschnitt "Referenz" dieses Handbuchs.
Konfigurieren von IPSec zwischen Anwendungsserver und Datenbankserver	Siehe "Vorgehensweise: Verwenden von IPSec zum Sichern der Kommunikation zwischen zwei Servern" im Abschnitt "Referenz" dieses Handbuchs.

Analyse

- Die Formularauthentifizierung ist in diesem Szenario ideal, da die Benutzer über keine Windows-Konten verfügen. Auf der Formularanmeldeseite werden die Anmeldeinformationen der Benutzer eingegeben. Die Überprüfung der Anmeldeinformationen muss durch den Anwendungscode durchgeführt werden. Dabei kann ein beliebiger Datenspeicher verwendet werden. Meist wird eine SQL Server-Datenbank verwendet, obwohl Active Directory eine Alternative zum Speichern der Anmeldeinformationen bereitstellt.
- Die webbasierte Anwendung wird als lokales Konto **ASPNET** mit möglichst geringen Rechten ausgeführt, sodass ein mögliches Risiko einer Offenlegung gemindert wird.
- Die URL-Autorisierung im Webserver ermöglicht nicht authentifizierten Benutzern, unbeschränkte Webseiten zu durchsuchen, und erzwingt eine Authentifizierung für beschränkte Seiten.
- Da keine Identitätswechsel aktiviert sind, erfolgt jeder Zugriff auf lokale oder Remoteressourcen durch die webbasierte Anwendung im Sicherheitskontext des Kontos **ASPNET**. Die ACLs müssen entsprechend konfiguriert werden.
- Die Anmeldeinformationen der Benutzer werden anhand einer benutzerdefinierten SQL Server-Datenbank überprüft. In der Datenbank werden Kennworthashes (mit Salt-Wert) gespeichert. Weitere Informationen finden Sie unter "Authentifizieren von Benutzern anhand einer Datenbank" in Kapitel 12, "Datenzugriffssicherheit".
- Durch die Windows-Authentifizierung bei SQL Server vermeiden Sie das Speichern der Anmeldeinformationen in Dateien auf dem Anwendungsserver und deren Übertragung über das Netzwerk.
- Die Verwendung eines duplizierten Windows-Kontos im Datenbankserver (das mit dem Enterprise Services-Prozesskonto übereinstimmt) führt zu einem gesteigerten Verwaltungsaufwand. Wenn das Kennwort auf einem Computer geändert wird, muss es synchronisiert und auf allen Computern aktualisiert werden. In einigen Szenarien können Sie eventuell ein Domänenkonto mit möglichst geringen Rechten verwenden, um die Verwaltung zu vereinfachen.
- Wenn die Webanwendung den Webdienst aufruft, muss der Webdienstproxy mithilfe von **DefaultCredentials** (d. h. dem ASP.NET-Prozesskonto **ASPNET**) konfiguriert werden.

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

  Weitere Informationen finden Sie unter "Übergeben von Anmeldeinformationen an Webdienste zur Authentifizierung" in Kapitel 10, "Webdienstsicherheit".
- SSL zwischen dem Webserver und der Webdienstschicht (die vor den Serviced Components auf dem Anwendungsserver liegt) stellt den Schutz der zwischen den beiden Servern gesendeten Daten sicher.

- Die Enterprise Services-Anwendung ist für eine rollenbasierte Sicherheit auf Anwendungsebene konfiguriert. Die Konfiguration lässt nur den Zugriff des lokalen **ASPNET**-Kontos (mit dem der Webdienst ausgeführt wird) auf die Serviced Components zu.
- IPSec zwischen Anwendungsserver und Datenbankserver stellt den Schutz der Daten sicher, die von und zu der Datenbank gesendet werden.
- SSL zwischen Browser und Webserver schützt die Anmeldeinformationen und Bankkontendetails.

Fallstricke

Die Anwendung muss die Identität des ursprünglichen Aufrufers an die Datenbank übertragen, damit die Anforderungen der Überwachung unterstützt werden. Die Identität des Aufrufers kann mithilfe von Parametern gespeicherter Prozeduren übergeben werden.

Verwandte Szenarien

Formularauthentifizierung anhand Active Directory

Die Anmeldeinformationen der Benutzer, die von der Formularanmeldeseite übernommen werden, können anhand einer Vielzahl von Speichern authentifiziert werden. Active Directory bildet eine Alternative zur Verwendung einer SQL Server-Datenbank.

Weitere Informationen

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

Verwenden von DCOM

Windows 2000 (SP3 oder SP2 mit QFE 18.1) oder Windows Server 2003 ermöglichen die Konfiguration von Enterprise Services-Anwendungen für die Verwendung eines statischen Endpunktes. Wenn ein Firewall den Client vom Server trennt, bedeutet das, dass in dem Firewall nur zwei Ports geöffnet werden müssen. Insbesondere muss Port 135 für RPC und ein Port für die Enterprise Services-Anwendung geöffnet werden.

Diese Erweiterung von DCOM ermöglicht die Auswahl von DCOM als Kommunikationsprotokoll zwischen dem Webserver und dem Anwendungsserver. Die Notwendigkeit einer Webdienstschicht ist in diesem Fall nicht gegeben.

Wichtig: Wenn die Anwendung die Übertragung verteilter Transaktionen zwischen den beiden Servern erfordert, muss DCOM verwendet werden. Transaktionen können nicht über SOAP übertragen werden. Im SOAP-Szenario müssen die Transaktionen von den Serviced Components auf dem Anwendungsserver initiiert werden.

Weitere Informationen

Weitere Informationen finden Sie in Kapitel 9, "Enterprise Services-Sicherheit".

Verwenden von .NET Remoting

Remoting kommt in Frage, wenn keine Dienste benötigt werden, die von den Enterprise Services bereitgestellt werden, wie z. B. Transaktionen, Warteschlangenkomponenten, Objektpooling usw. .NET Remoting-Lösungen unterstützen ebenfalls Netzwerklastenausgleich auf der mittleren Ebene. Beachten Sie Folgendes, wenn Sie .NET Remoting verwenden:

- Verwenden Sie zwecks optimaler Leistung den TCP-Kanal und Host in einem Windows-Dienst. Beachten Sie, dass dieser Kanal standardmäßig keinen Authentifizierungs- und Autorisierungsmechanismus bereitstellt. Der TCP-Kanal ist für Szenarien mit vertrauenswürdigen Subsystemen konzipiert. Sie können mithilfe einer IPSec-Richtlinie einen sicheren Kanal einrichten und sicherstellen, dass nur der Webserver mit dem Anwendungsserver kommuniziert.

- Wenn Authentifizierungs- und Autorisierungsprüfungen mit **IPrincipal**-Objekten erforderlich sind, sollten Sie die Remoteobjekte in ASP.NET verwalten und den HTTP-Kanal verwenden, sodass die IIS- und ASP.NET-Sicherheitsfeatures verwendet werden können.
- Das Remoteobjekt kann unter Verwendung der Windows-Authentifizierung Verbindungen zur Datenbank herstellen und die Prozessidentität des Hosts verwenden (entweder ASP.NET oder eine Windows-Dienstidentität).

Weitere Informationen

Weitere Details zur .NET Remoting-Sicherheit finden Sie in Kapitel 11, ".NET Remoting-Sicherheit".

Zusammenfassung

In diesem Kapitel wurde beschrieben, wie häufige Internetanwendungsszenarien gesichert werden.

Informationen für Intranet- und Extranetanwendungsszenarien finden Sie in Kapitel 5, "Intranetsicherheit", und Kapitel 6, "Extranetsicherheit".