Brokered Authentication: Security Token Service (STS)

Context

Web services need to authenticate clients in a heterogeneous environment so that additional controls such as authorization and auditing can be implemented. The organization has decided to use an *authentication broker* to provide a common access control infrastructure for a group of applications. The authentication broker negotiates trust between client applications and Web services; this removes the need for a direct relationship. The authentication broker should issue signed security tokens that can be used for authentication.

Problem

How does the Web service verify the credentials presented by the client?

Forces

Any of the following conditions justifies using the solution described in this pattern:

- **Clients requiring authentication are implemented on a variety of platforms within the organization, and interoperability is required between those platforms**. Using a standards based mechanism for authentication helps ensure interoperability between different platforms.
- **The organization has identified a need for security tokens that are extensible and include claims that support additional security functions**. The authentication broker must be flexible enough to receive and issue tokens that support additional functionality such as authorization, auditing, and custom authentication.

The following condition is an additional reason to use the solution:

- **The environment includes organizational boundaries that are protected by firewalls**. The authentication broker must be able to issue security tokens that can traverse these boundaries, including passing through ports that are commonly enabled on firewalls.

The following conditions are not resolved by the base pattern, but they are resolved by the extensions provided at the end of this pattern:

- **Users access multiple clients that call Web services, resulting in the need for single sign on (SSO) capabilities**. To ensure a positive user experience, users should have to enter a user name and password only when logging on to a workstation; users should not have to re-enter them multiple times when accessing multiple clients.
- **The environment includes multiple security domains**. Clients must be able to obtain security tokens, so that resources such as services can be accessed in a different security domain using a security token issued by the authentication broker in its own security domain.

Solution

Use brokered authentication with a security token issued by a Security Token Service (STS). The STS is trusted by both the client and the Web service to provide interoperable security tokens.

The client sends an authentication request, with accompanying credentials, to the STS. The STS verifies the credentials presented by the client, and then in response, it issues a security token that provides proof that the client has authenticated with the STS. The client presents the security token to the Web service. The Web service verifies that the token was issued by a trusted STS, which proves that the client has successfully authenticated with the STS.

The protocol used for issuing security tokens is based on WS-Trust. WS-Trust is a Web service specification that builds on WS-Security. It describes a protocol used for issuance, exchange, and validation of security tokens. WS-Trust provides a solution for interoperability by defining a protocol for issuing and exchanging security tokens, based on token format, namespace, or trust boundaries.

In WS-Trust, the type of message sent to an STS to request issuance of a security token is known as a Request Security Token (RST) message. The RST message contains credentials for the client to be authenticated, such as the user ID and password contained in a UsernameToken. The response message from the STS is known as a Request Security Token Response (RSTR) message. The RSTR contains a security token, such as an XML Security Assertion Markup Language (SAML). For more information about WS-Trust, see *Web Services Trust Language (WS-Trust)* on MSDN.

SAML Tokens

SAML (Security Assertion Markup Language) tokens are standards-based XML tokens that are used to exchange security information, including attribute statements, authentication decision statements, and authorization decision statements. SAML tokens are also extensible; this means you can extend the schema of the token to meet additional requirements.

SAML tokens are important for Web service security because they provide cross-platform interoperability and a means of exchanging security information between clients and services that do not reside within a single security domain. They can be used as part of an SSO solution allowing a client to talk to services running on disparate technologies.

The SAML specifications cover a broad range of topics — from the format of the actual SAML token to a protocol that can be used for token request and issuance. Microsoft products use the WS-* specifications, which include the use of SAML assertions but not the SAML protocol. Instead of the SAML protocol, token issuance and federation uses the WS-Trust and WS-Federation set of specifications. Currently, ADFS in Windows Server 2003 R2 uses SAML 1.1 tokens and the WS-Federation passive client profile specification to enable SSO scenarios within Web applications. For more information about ADFS, see Introduction to ADFS on Microsoft TechNet. Future support for active client scenarios (such as SSO support for Web services) is under development.

(continued)

SAML Tokens *(continued)*

The SAML standard is still evolving from version to version, and the versions are not currently interoperable. At the time of this writing, there is increasing adoption of the SAML 1.1 specification, but implementations may need to be modified to support future versions of the SAML standard if SAML tokens are used.

For more information about the SAML 1.1 specification, including the protocol for request and issuance of SAML tokens, see the OASIS Web site.

Participants

Brokered authentication with STS involves the following participants:

- **Client**. The client accesses the Web service. The client provides the credentials for authentication during the request to the Web service.
- **STS**. The STS is the Web service that authenticates clients by validating credentials that are presented by a client. The STS can issue to a client a security token for a successfully authenticated client.
- **Service**. The service is the Web service that requires authentication of a client prior to authorizing the client.

Process

Figure 1.11 illustrates the process by which a security token is issued to the client by the STS and then used to authenticate with a service, which then returns a response to the client.

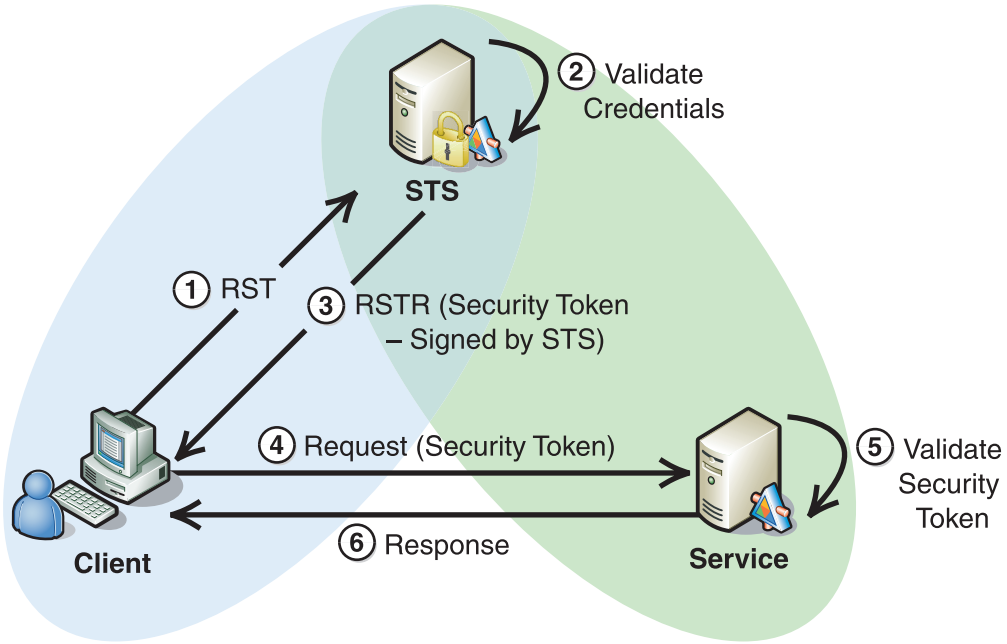

Figure 1.11

STS token issuance and request/response

As illustrated in Figure 1.11, the following steps describe STS token issuance and request/response process:

1. **The client initializes and sends authentication request to the STS**. The authentication request to the STS is in the form of an RST message. This step can be performed by presenting the client's identifier and proof-of-possession (such as user name and password) directly to the STS or by using a token issued by an authentication broker (such as an X.509 digital signature or Kerberos tokens).

   The RST message contains a security token that holds the client's credentials, which are required to authenticate the client. Claims in the client's credentials, such as a password, may be sensitive in nature, so it is very important to secure the RST. The specific security mechanism used for securing the RST depends on the relationship between the client and the STS. For example, the client and STS may use Kerberos tokens or X.509 certificates to sign and encrypt messages sent between them. For more information about securing messages, see Data Confidentiality and Data Origin Authentication in Chapter 2, "Message Protection Patterns."
2. **The STS validates the client's credentials**. After the STS determines that the client's credentials are valid, it may also decide whether to issue a security token for the authenticated client. For example, the STS may have a policy where it issues tokens only for users who belong to a specific role or for valid X.509 certificates that can be validated through a specific trust chain.
3. **The STS issues a security token to the client**. If the client's credentials are successfully validated, the STS issues a security token (such as a SAML token) in an RSTR message to the client; typically, the security token contains claims related to the client. The security token is usually signed by the STS; when the security token is signed by STS, the service can confirm that the token was issued by the STS and that the security token was not tampered with after it was issued.
4. **The client initializes and sends a request message to the service**. After the client receives a security token from the STS, it initializes a request message that includes the issued security token, and then it sends the request message to the service.
5. **The service validates the security token and processes the request**. The security token is validated by the service to verify that the token was issued by the trusted STS and that the token was not tampered with after it was issued. After the token is validated by the service, it is used to establish security context for the client, so the service can make authorization decisions or audit activity.
6. **(Optional) The service initializes and sends a response message to the client**. A response is not always required. Frequently, the response message contains sensitive data, so it should be secured.

A client may also specify the scope of the request for a security token to the STS. Scope is a value that identifies the target of the client; it can be as granular as a single operation of the Web service or as broad as an application domain. The token issued by the STS can contain usage constraints that correspond to the scope of the request.

Scope can be used to provide resource level authorization, with the STS comparing the value in the scope to a list of clients that are authorized to access the target.

Resulting Context

This section describes some of the more significant benefits, liabilities, and security considerations of using this pattern.

Note: The information in this section is not intended to be comprehensive. However, it does discuss many of the issues that are most commonly encountered for this pattern.

Benefits

The benefits of using the Brokered Authentication: Security Token Service (STS) pattern include the following:

- This pattern provides a flexible solution for exchanging one type of security token for another to accomplish a variety of goals in a Web service environment, such as authentication, authorization, and exchanging session keys.
- The solution is not dependent on any one mechanism, such as the Kerberos protocol or X.509 to secure messages. This makes it easier to enable different authentication protocols to interoperate, by adding a level of abstraction on top of existing protocols.

Liabilities

The layer of abstraction provided by the STS means that the STS must use another underlying security protocol to provide functionality such as authentication and authorization. This can make the STS a more difficult solution to implement, particularly in cases where a custom solution is used.

Security Considerations

Security considerations associated with the Brokered Authentication: Security Token Service (STS) pattern include the following:

- Request and response messages between the client and the STS often contain sensitive information, such as user passwords and session encryption/signature keys, so they should be protected using data encryption and data origin authentication. For more information about data encryption, see Data Confidentiality in Chapter 2, "Message Protection Patterns." For more information about data origin authentication, see Data Origin Authentication in Chapter 2, "Message Protection Patterns."
- Request and response messages between the client and the STS and between the client and the service may also be susceptible to message replay attacks if communication is secured at the message layer. For information about preventing an attacker from replaying messages, see Message Replay Detection in Chapter 5, "Service Boundary Protection Patterns."

Extensions

The extensions described here build on the base pattern to provide additional capabilities.

Extension 1 — Establishing a Secure Conversation

This extension can be used to establish a secure conversation with the STS. There are several reasons for establishing a secure conversation with the STS, including:

- Preventing the client from having to present a user name and password each time it accesses a different service. This could involve the client having to cache the client's original credentials (which is not considered a safe security practice) or prompting users to provide their credentials each time.
- Improving performance when resource-intensive forms of credentials, such as X.509 digital signatures, are used. Creation and validation of X.509 digital signatures is a computationally intensive process, so performance can be improved if they are used less frequently.

In this extension, the client obtains a Security Context Token (SCT) (which demonstrates that the client has been authenticated) from the STS and caches it. After the client is authenticated with the STS, the client can use the session token to request a service token for communication with a service. The way the STS validates a security token presented by a client and issues service tokens is similar to how the Kerberos protocol validates a ticket-granting ticket and issues a service ticket. For more information about the Kerberos protocol, see Brokered Authentication: Kerberos in Chapter 1, "Authentication Patterns." Figure 1.12 illustrates this behavior.

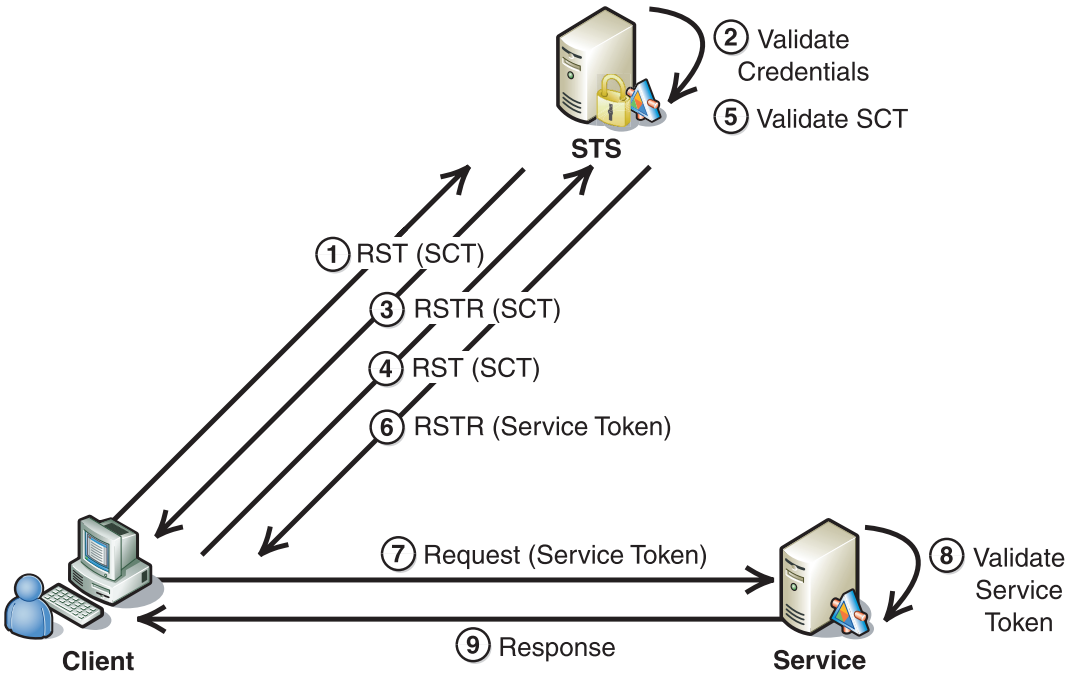

Figure 1.12

Establishing a secure conversation with the STS

This extension is based on the use of WS-SecureConversation to establish a session between the client and the service. WS-SecureConversation is a Web service specification that builds on WS-Security and WS-Trust. It describes how to establish a lightweight security context between two parties. The security context uses session keys; these session keys become the basis for encrypting and signing subsequent message exchanges, which results in more efficient secure communications between the two parties. For more information about WS-SecureConversation, see *Web Services Secure Conversation Language (WS-SecureConversation)* on MSDN.

Note: The security of any conversation depends on the key exchange mechanism. Typically, the key exchange mechanism is based on a key management infrastructure, such as one based on PKI or shared secrets.

The secure conversation extension can also be applied to the other direct authentication and brokered authentication patterns to optimize interactions between two parties. In this instance, the secure conversation is applied to demonstrate how a session can be established between a client and an STS.

While the establishment of a secure conversation with the STS logically includes three parties — the client, the STS and the service — the solution is typically implemented with the STS residing on the same node as the service. The STS issues SCTs to maintain state between the client and the STS, and it also issues service tokens for communication between the client and the service. The client can use the service token to authenticate with the service, and may even establish a secure conversation with the service.

This extension builds on the base pattern to provide additional capabilities. In addition to resolving the forces stated for the base pattern, it also resolves the following condition:

- **Users access multiple clients that call Web services, resulting in the need for single sign on (SSO) capabilities**. To ensure a positive user experience, users should have to enter a user name and password only when logging on to a workstation; users should not have to re-enter them multiple times when accessing multiple clients.

Process

This section describes the steps of the process illustrated in Figure 1.12. It demonstrates how the STS issues Security Context Tokens (SCTs) to allow the client to establish a session with STS. An SCT is a lightweight security token used to gain access to the STS and to optimize secure communications between the client and the STS.

Ad illustrated in Figure 1.12, the following steps describe the STS process:

1. **The client initializes and sends an authentication request to the STS**. The authentication request to the STS is in the form of an RST message. This step can be performed by presenting the client's identifier and proof-of-possession (such as user name and password) directly to the STS or by using a token issued by an authentication broker (such as an X.509 digital signature or Kerberos protocol Token). Information, such as a password, is sensitive in nature, so it is very important to secure the RST using message protection. For more information see Chapter 2, "Message Protection Patterns."
2. **The STS validates the client's credentials**. The client's credentials are a series of claims that prove the client's identity or confirm that the client has successfully authenticated with another trusted authentication broker such as an X.509 CA, a Kerberos KDC, or another STS.

3. **The STS issues a Security Context Token (SCT) to the client**. The SCT can be used by the client each time additional security tokens are required, instead of the client presenting the client's original credentials each time. The scope of the issued SCT is limited to the STS regardless of whether the client specified the scope in the initial RST. This prevents the client from using the SCT to directly access a service.
4. **The client caches the SCT**. By caching the SCT, the client can establish a session with the STS. Then the client can make subsequent requests to the STS without having to present the client's original credentials each time. The STS can include claims about the client in the SCT, which the STS can use to make authorization decisions.
5. **The client requests a service token from the STS to communicate with the service**. When the client attempts to communicate with a specific service, it sends another RST to the STS. This RST contains the SCT that was initially issued by the STS for the authenticated client. In the RST, the client specifies the target Web service as the scope of the request.
6. **The STS responds to the service token request**. The STS may have established a policy to determine whether the client is authorized to access the service that is specified in the scope. If the client is allowed access to that service, the STS issues to the client a security token that is used to authenticate with the service.
7. **The client initializes and sends request message to the service**. After the client obtains the required security token from the STS, it initializes a request message that includes the issued service token, and sends it to the service.
8. **The service validates service token and processes the request**. The service ensures that the security token was issued by the trusted STS and that the token was not tampered with after it was issued. After the token is validated by the service, it is used to establish a security context for the client, so the service can make authorization decisions or audit activity.
9. **Service initializes and sends response message to the client**. The client may not always expect a response from the service. The client knows whether to expect a response from the service if it has been specified in the Web service contract using Web Service Discovery Language (WSDL).

Note: It is also possible for the client to establish a secure conversation with the service. In this case, Step 7 would be preceded by a request for a SCT from the service using the newly issued service token (for example, a SAML token) as the basis for the initial authentication with the service.

Extension 2 — Web Service Federation

A client may need to communicate with Web services that operate across organizational boundaries. Typically, the different organizations each have their own autonomous security domains established. In this situation, the client is authenticated in the security domain where the client operates, but it must be authorized and audited within the security domain where the service operates for the client to be able to call the service.

This extension builds on the base pattern to provide additional capabilities. In addition to resolving the forces stated for the base pattern, it also resolves the following condition:

- **The environment includes multiple security domains**. Clients must be able to obtain security tokens, so that resources such as services can be accessed in a different security domain using a security token issued by the authentication broker in its own security domain.

With security federation, security claims can be propagated and consumed across different security domains to support identification, authentication, authorization, and auditing.

After the client is authenticated, the token obtained from the STS is exchanged for a token that is useable in the target security domain. Security domains can be federated in different ways, depending on the operating environment and the security requirements for applications within the federation.

Note: This extension demonstrates at a high level how a SAML STS can be used as part of a larger federation solution. As such, a comprehensive discussion of federation is outside the scope of this pattern. The federation solution described here would include support for additional capabilities, such as mapping role information from one domain into equivalent role information in another domain to provide support for authorization of the client.

Figure 1.13 illustrates an example of interaction between a client and a service in two different security domains that participate in a federation through their respective STSs:

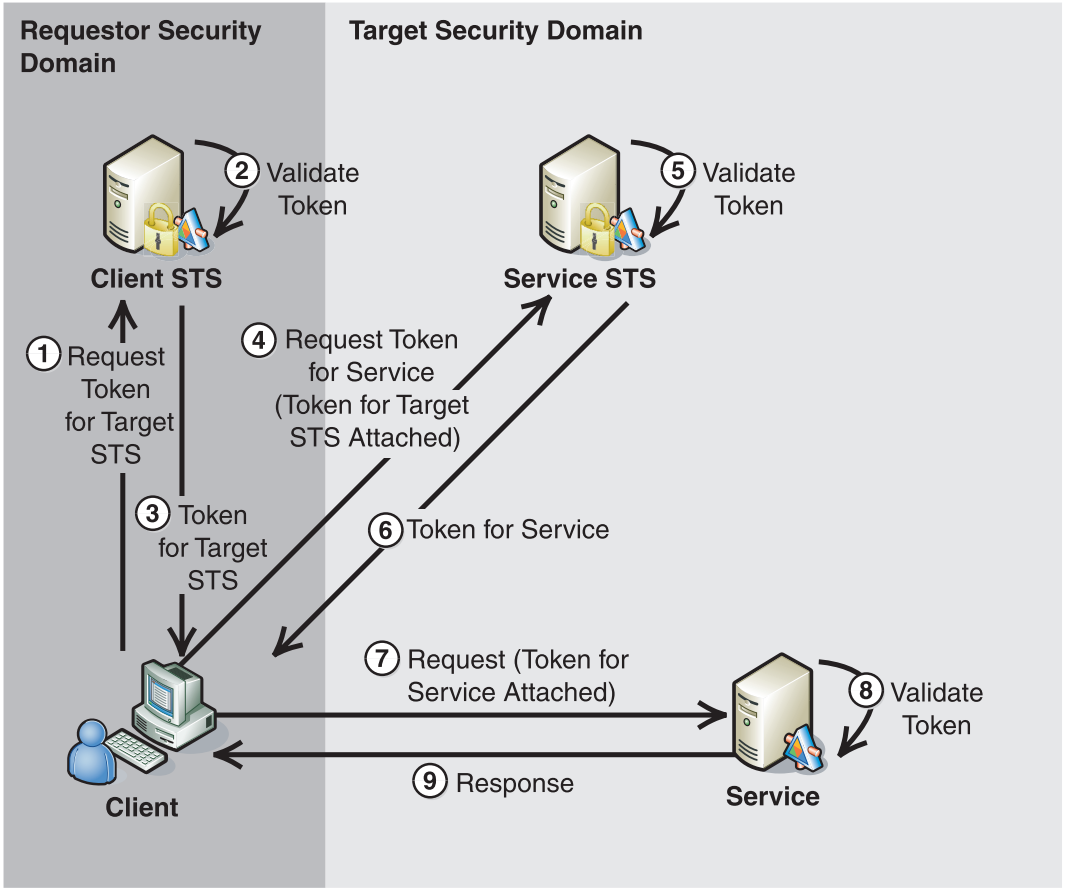

Figure 1.13

Obtaining a security token to authenticate with a service in a different security domain

Note: In this model of federation, the client is responsible for requesting the appropriate security token, which is consumable by the target Web service, as shown in Figure 1.13. This example can be used for active or passive clients. Support for passive clients is possible if the STS issues security tokens that are useable through HTTPS and can be cached by the browser.

As illustrated in Figure 1.13, the following steps describe Web service federation process:

1. **The client requests a security token to communicate with the STS in the target security domain**. The client presents authentication credentials, a security token previously issued by the STS, or a security token issued by another trusted authentication broker to obtain the security token for the target security domain.
2. **The STS in the client's domain validates the credentials or security token presented by the client**. The STS in the client's security domain may make authorization decisions about whether to issue a security token to the client for use in the security domain where the target service operates.
3. **The STS in the client's domain issues a security token to the client that is used to obtain a service token from the STS in the domain where the target service operates**. If the client is authenticated and authorized by the STS in the client's domain that STS issues a security token to the client to communicate with the STS in the target domain where the Web service operates. Based on the target security domain, the STS in the client's domain knows the type and scope of security token to issue.
4. **The client requests a security token from the STS in the target security domain**. The client presents the token issued by its STS to communicate with the STS of the target security domain.
5. **The target STS validates the client's security token**. The STS in the target security domain verifies that the token presented by the client originated from an STS in a trusted security domain. After the STS in the Web service's security domain validates the security token presented by the client, it may make an authorization decision about whether the client is authorized to access the requested service.
6. **The target STS issues a security token to communicate with the service**. If the target STS decides that the request is valid and the client is authorized to communicate with the service, it will issue a security token to the client that can be used to communicate with the service.
7. **The client sends a request message to the service**. The client attaches the security token it received from the STS in the target service's security domain to the request and sends it to the service.
8. **The service validates the security token attached to the request**. The service verifies that the token presented by the client was issued by a trusted STS.
9. **The service initializes and sends a response message to the client**.

Related Patterns

Three types of patterns are related to this pattern: parent patterns, child patterns, and alternate patterns.

The following parent pattern is related to the Brokered Authentication: Security Token Service (STS) pattern:

- **Brokered Authentication**. This pattern describes how to prove a client's identity to an authentication broker so that the broker can issue a security token.

The following child pattern is related to the Brokered Authentication: Security Token Service (STS) pattern:

- **Implementing Message Layer Security with a Security Token Service (STS) in WSE 3.0**. This pattern provides implementation guidelines for using an STS in WSE 3.0 to implement brokered authentication. This pattern is currently still in development, and is scheduled for completion in early 2006.

The following alternate patterns are related to the Brokered Authentication: Security Token Service (STS) pattern:

- **Brokered Authentication: Kerberos**. The Kerberos protocol provides an alternative to X.509 based on the Kerberos authentication protocol.
- **Brokered Authentication: X.509 PKI**. This pattern describes a specialized authentication broker based on the X.509 PKI standard.

More Information

For more information about authorization on the .NET Framework, see "Authentication and Authorization" in *Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication* on MSDN: *http://msdn.microsoft.com/practices/Topics/security/default.aspx?pull=/library/en-us/dnnetsec/html/SecNetch03.asp*.

For more information about Web services security, see OASIS Standards and Other Approved Work (including WS-Security) on the OASIS Web site: *http://www.oasis-open.org/*.

For more information about the Kerberos protocol specifications, see RFC 1510: The Kerberos Network Authentication Service (V5): *http://www.faqs.org/rfcs/rfc1510.html*.

For more information about Kerberos authentication in Windows Server 2003, see "Kerberos Authentication Technical Reference" on Microsoft TechNet: *http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/TechRef/b748fb3f-dbf0-4b01-9b22-be14a8b4ae10.mspx*.

For a general overview of PKI technologies, see "PKI Technologies" on Microsoft TechNet: *http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/TechRef/6d5d9ef3-75ca-46c1-acf6-57dc7e9a6adf.mspx*.

For more information about WS-Trust, see *Web Services Trust Language (WS-Trust)* on MSDN: *http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-trust.pdf*.

For more information about ADFS, see "Introduction to ADFS" on Microsoft TechNet: *http://technet2.microsoft.com/WindowsServer/en/Library/c67c9b41-1017-420d-a50e-092696f40c171033.mspx*.

For more information about Security Assertion Markup Language (SAML), go to the OASIS Web site: *http://www.oasis-open.org/specs/index.php#samlv1.1*.

For more information about WS-SecureConversation, see *Web Services Secure Conversation Language (WS-SecureConversation)* on MSDN: *http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-secureconversation.pdf*.

For more information about SAML token profile 1.0, see *Web Security Services: SAML Token Profile* on the Oasis Web site: *http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf*.