By assigning the client certificate's SHA1 thumbprint to the security token identity, you can use the hexadecimally encoded value of the client certificate's SHA1 thumbprint in the service's **<authorization>** assertion instead of the subject distinguished name. You can use the Certificates MMC Snap-In tool to obtain the hexadecimally encoded SHA1 thumbprint for a certificate. The following XML example provides an example of how to use the client certificate's SHA1 thumbprint in an **<authorization>** assertion in the service's policy cache.

```
...
<authorization>
      <allow user="ca7601381b4578502b62b8809825664f1e78dfa2" />
      <deny user="*" />
</authorization>
...
```

This code example mitigates the risk of confusing client identities by providing a way to identify client certificates that is more likely to be unique, as it performs authorization on the service when CAs issue different certificates with the same subject distinguished name.

Implementing Message Layer Security with a Security Token Service (STS) in WSE 3.0

Note: This pattern is currently under development. It is due for release in early 2006.

Context

You are implementing brokered authentication in an application deployed on computers running Windows operating system software with security implemented at the message layer. Web services need to authenticate clients in a heterogeneous environment so that you can implement additional controls, such as authorization and auditing. The authentication broker negotiates trust between client applications and Web services, which removes the need for a direct relationship. The authentication broker should issue signed security tokens for authentication.

Implementation Strategy

A QuickStart that demonstrates how to develop a Web Service Enhancements (WSE) 3.0 Security Token Service (STS) that issues XML tokens is currently under development. This pattern will be updated when the QuickStart is released.

If you are interested in obtaining a Community Technical Preview (CTP) release or would like to contribute requirements, join the Security Token Service Quickstart community workspace.

References for Transport Layer Security

There is a lot of good information available on using transport layer security to secure Web services, so this information is provided in the form of the following references, which point you to appropriate guidance for implementing transport layer security. It contains the following sections:

- Implementing Brokered Authentication Using Windows Integrated Security on IIS
- Implementing Transport Layer Data Confidentiality Using HTTPS
- Implementing Transport Layer Security Using HTTP Basic over HTTPS
- Implementing Transport Layer Security Using X.509 Certificates and HTTPS
- Implementing Transport Layer Security with Kerberos and IPSec on Windows Server 2003

Implementing Brokered Authentication Using Windows Integrated Security on IIS

This implementation reference provides guidance for implementing brokered authentication on an existing Kerberos version 5 protocol infrastructure at the transport layer. Brokered authentication using Windows Integrated Security on Internet Information Services (IIS) 6.0 allows you to call applications and Web services to validate credentials against an Active Directory domain controller, as an implementation of the Kerberos protocol. The calling applications and Web services can validate credentials against the same Active Directory domain or multiple Active Directory domains joined by a cross-domain trust relationship. The Web services also can impersonate the caller to access resources controlled under a trusted Active Directory domain.

To implement brokered authentication using Windows Integrated Security on IIS 6.0, you must perform the following tasks:

1. Implement transport-layer brokered authentication using Windows Integrated Security.
2. Configure IIS 6.0 to require Windows Integrated Security.
3. Add the credentials for the client that the Web service will authenticate to the credential cache of the Web service proxy that communicates with the Web service.

The benefits to this approach include:

- A minimal amount of code and configuration work for the implementation.
- When you implement this approach using Kerberos authentication instead of NTLM authentication, you can use it to flow the caller's identity across multiple system hops.

One liability to this approach is that firewall boundaries may not allow Kerberos authentication traffic between the calling application and the Kerberos Key Distribution Center (KDC) or between the Web service and the Kerberos KDC.

It is also important to take into account that this approach does not provide data confidentiality or data origin authentication for messages sent between the calling application and the Web service. Use HHTPS or IPSec to secure messages between the calling application and Web service. For more information about these limitations, see, "Implementing Transport Layer Security Using HTTP Basic over HTTPS" and "Implementing Transport Layer Security Using Kerberos and IPSec on Windows Server 2003."

For more information about implementing this approach, see the following resources:

- To learn more about Windows Integrated Security, see the "Authentication and Authorization Strategies" section in "Web Services Security" on MSDN.
- To call a Web service configured to use Windows Integrated Authentication, see the "Passing Credentials for Authentication to Web Services" section in "Web Services Security" on MSDN.

Implementing Transport Layer Data Confidentiality Using HTTPS

This implementation reference provides guidance on implementing data confidentiality using X.509 certificates at the transport layer. To implement data confidentiality using X.509 certificates at the transport layer, you must perform the following tasks:

1. Implement transport layer security using Secure Sockets Layer (SSL).
2. Configure the Web service virtual directory in IIS 6.0 to require SSL.

One benefit to this approach is that SSL is a well-established protocol that is easy to configure and implement on the Windows platform. However, there are several liabilities and security considerations to take into account with this approach. You can only establish SSL point-to-point as opposed to end-to-end, as message layer security is capable of doing. Additional liabilities of this approach include:

- Communication between several points configured for SSL rather than end-to-end at the message layer may cause unacceptable application response times.
- All points in the communication must be sufficiently trusted to establish SSL.

In some cases, these liabilities may warrant using a different approach, such as implementing message layer X.509 security using X509Security Tokens in WSE 3.0.

This approach has the following security considerations:

- Vulnerabilities exist in previous versions of SSL. The server and client need to have security patches installed on them to mitigate known vulnerabilities.
- Microsoft strongly recommends configuring IIS 6.0 to require strong (128-bit) SSL encryption for increased protection of data confidentiality.

For more information about implementing this approach, see the following resources:

- To learn how an SSL session is established between two parties, see "Description of the Secure Sockets Layer (SSL) Handshake" on Microsoft Help and Support: *http://support.microsoft.com/default.aspx?scid=kb;%5bLN%5d;Q257591*.
- To learn how to implement SSL, see:
  - "How To Set Up SSL on a Web Server" on MSDN: *http://msdn.microsoft.com /library/default.asp?url=/library/en-us/secmod/html/secmod30.asp*.
  - "How To Call a Web Service Using SSL" on MSDN: *http://msdn.microsoft.com /library/default.asp?url=/library/en-us/secmod/html/secmod28.asp*.
  - "How To Call a Web Service Using Client Certificates from ASP.NET" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmod /html/secmod27.asp*.

Implementing Transport Layer Security Using HTTP Basic over HTTPS

This implementation reference provides guidance on implementing direct authentication using HTTP Basic over HTTPS on the transport layer. An advantage of this implementation is that it can make use of an existing infrastructure.

To implement transport layer security using HTTP Basic over HTTPS, you must perform the following tasks:

1. Implement transport layer direct authentication using HTTP basic authentication.
2. Configure IIS 6.0 to require HTTP basic authentication for the virtual directory hosting the service.
3. On the client, add the client's credentials to the credential cache of the proxy that communicates with the service.

This approach is generally considered easy to configure and simple to use. It uses a well established and widely supported type of direct authentication in a Web environment. However, one liability to this approach is that it provides no message protection capabilities. Using HTTP basic authentication, the client's credentials are passed in plaintext in transit, which makes them easily susceptible to eavesdropping by an attacker. Therefore, Microsoft strongly recommends using SSL to provide data confidentiality to prevent eavesdropping attacks against the credentials.

For more information about implementing this approach, see the following resources:

- To learn how to configure IIS for HTTP basic authentication, see "Basic Authentication in IIS 6.0" on Microsoft TechNet: *http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/abbca505-6f63-4267-aac1-1ea89d861eb4.mspx*.
- To learn how to call a Web service that requires credentials, see the "Passing Credentials for Authentication to Web Services" section in "Web Services Security" on MSDN: *http://msdn.microsoft.com/library/en-us/secmod/html/secmod10.asp*.

Implementing Transport Layer Security Using X.509 Certificates and HTTPS

This implementation reference provides guidance for implementing brokered authentication using X.509 certificates on the transport layer. Transport layer security using X.509 certificates and HTTPS secures point-to-point communication. Messages do not require intermediaries to process them and they are not securely persisted for any period of time.

To implement transport layer security using X.509 certificates and HTTPS, you must perform the following tasks:

1. Implement transport layer security using SSL.
2. Configure the Web service virtual directory to use SSL and require client certificates.

This approach has the following benefits:

- It provides brokered authentication, data confidentiality, and data origin authentication capabilities in one solution.
- It uses SSL, which is a well established protocol that is easy to configure and implement on the Windows platform.

The disadvantage of this approach is that you can only establish point-to-point SSL, not end-to-end as message layer security is capable of doing. There are certain liabilities as a result of using SSL that may warrant you to use a different approach, such as implementing message layer X.509 security using X509Security Tokens with WSE 3.0. These liabilities include:

- Communication between several points configured for SSL rather than end-to-end at the message layer may cause unacceptable application response times.
- All points in the communication must be sufficiently trusted to establish SSL.

This approach has the following security considerations:

- Vulnerabilities exist in previous versions of SSL. The server and client need to have security patches installed to mitigate known vulnerabilities.
- Microsoft strongly recommends configuring IIS 6.0 to require strong (128-bit) SSL encryption for increased protection of data confidentiality.

For more information about implementing this strategy, see the following resources:

- To learn about how an SSL session is established between two parties, see "Description of the Secure Sockets Layer (SSL)" on Microsoft Help and Support: *http://support.microsoft.com/default.aspx?scid=kb;%5bLN%5d;Q257591*.
- To learn about how a client authenticating to a service using SSL operates, see "Description of the Client Authentication Process During the SSL Handshake" on Microsoft Help and Support: *http://support.microsoft.com/kb/257586/EN-US/*.
- To learn about how to implement SSL, see the following documentation:
  - "How To Set Up SSL on a Web Server" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmod/html/secmod30.asp*.
  - "How To Call a Web Service Using SSL" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmod/html/secmod28.asp*.
  - "How To Call a Web Service Using Client Certificates from ASP.NET" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmod/html/secmod27.asp*.

Implementing Transport Layer Security with Kerberos and IPSec on Windows Server 2003

This implementation reference provides guidance on how to implement data confidentiality and data origin authentication using the Kerberos protocol and IPSec on Windows Server 2003 at the transport layer. The solution provides data confidentiality and data origin authentication between two servers hosting Web services, and another resource, such as an application server or a database.

This approach secures point-to-point communication. Messages do not require intermediaries to process them and they are not securely persisted for any period of time. Data origin authentication is done at the host layer instead of at the application or user layer. The two hosts that require data confidentiality and data origin authentication are joined to the same Kerberos realm or to different Kerberos realms that have established a cross-trust relationship.

To implement transport layer security with the Kerberos protocol and IPSec on Window Server 2003, you must perform the following tasks:

1. Implement network layer security using IPSec.
2. Configure IPSec send-and-receive policies to send and receive messages on each host to communicate with the other host that requires data confidentiality and data origin authentication.
3. Configure IPSec to use Kerberos mode authentication.

The benefits to this approach include the following:

- It provides data confidentiality and data origin authentication capabilities in one solution.
- IPSec is a well established protocol that is easy to configure and implement on the Windows Server 2003 platform.
- IPSec has very good performance compared to other solutions for data confidentiality and data origin authentication because it is below the protocol layer in the network stack.

One liability of this approach is that IPSec does not exercise very fine control over how it uses the Kerberos protocol to authenticate with another host. If business requirements exist for auditing or data origin authentication at the user or application layer, another mechanism other than IPSec must provide it.

For more information about IPSec and how to deploy it on Windows Server 2003, see "IPSec" on Microsoft.com: *http://www.microsoft.com/windowsserver2003/technologies/networking/ipsec/default.mspx*.

More Information

For information about Web Services Security, see "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)": *http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf*.

For information about derived key tokens, see "Web Services Secure Conversation Language (WS-SecureConversation)": *http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf*.

For information about how to configure a **SqlMembershipProvider**, see "How To: Use Membership in ASP.NET 2.0" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag2/html/PAGHT000022.asp*.

For information about creating a custom ASP.NET 2.0 membership provider, see "Building Custom Providers for ASP.NET 2.0 Membership" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnaspp/html/bucupro.asp*.

For information about configuring WSE 3.0 to prevent replay attacks, see "Web Services Enhancements 3.0 <replayDetection> Element" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/b4fa188d-4804-40bd-877b-c01058555013.asp*.

For more information about performance objectives, see "Improving .NET Performance and Scalability" on MSDN: *http://msdn.microsoft.com/practices/Topics/perfscale/default.aspx?pull=/library/en-us/dnpag/html/scalenet.asp*.

For information about WSE 3.0 policy, see "Securing a Web Service" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/7b8f29da-22d5-4e03-b645-15011a80e548.asp*.

For information about Kerberos assertion policy settings, see "<kerberosSecurity> Element" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/bde6a6dd-00e4-4c37-aa8d-8821f2f25bc5.asp*.

For more information about performance objectives see, "Improving .NET Performance and Scalability" on MSDN: *http://msdn.microsoft.com/practices/Topics/perfscale/default.aspx?pull=/library/en-us/dnpag/html/scalenet.asp*.

For information about installing X.509 certificates in the local certificate store, see "How to: Use the X.509 Certificate Management Tools" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse/html/21eb7fb5-bd11-4cce-be0c-7b3d0cd14acb.asp?frame=true*.

For information about how to install X.509 certificates in the local machine certificate store, see "Certificates How To" on Microsoft TechNet: *http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/ServerHelp/fb037b9f-8956-411c-a3e8-ce1dfe37da11.mspx*.

For more information about configuring the behavior of X.509 security in WSE 3.0, see "<x509> Element" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/72b7b9c9-63dd-4ce7-a25f-e40b164912d2.asp* in the WSE documentation.

For information about how to set the **findType** and **findValue** attributes for the **<x509>** element, see "<x509> Element (Policy)" in the WSE 3.0 documentation on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/4caad727-778e-4c57-90f8-0edca69eed1f.asp*.

For information about configuring other settings for this policy assertion, see "<mutualCertificate10> Element" in the WSE 3.0 documentation on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/973d38d8-6347-4617-983f-089e64a2b02c.asp.*

For more information about performance objectives, see "Improving .NET Performance and Scalability" on MSDN: *http://msdn.microsoft.com/practices/Topics/perfscale/default.aspx?pull=/library/en-us/dnpag/html/scalenet.asp*.