Authorization in Windows Communication Foundation

Windows Communication Foundation (WCF) will integrate seamlessly with the role-based security features that are built into the .NET Framework. WCF will communicate the sender's credentials to the receiving code using the usual **Thread.CurrentPrincipal** property. Because of this, you can perform authorization either declaratively using **PrincipalPermissionAttribute** or imperatively using **IPrincipal.IsInRole**.

WCF will also incorporate a sophisticated claims-based authorization infrastructure exposed through an object named "authorization context." This allows more complex authorization decisions to be made based on additional claims provided in tokens within an incoming message.

Guidance for authorization using WCF will be incorporated in an updated version of this guidance that will also incorporate implementations using WCF.

For a complete description of authorization on the .NET Framework, see Authentication and Authorization on MSDN®.

The remainder of this chapter contains the architecture and design patterns related to authentication. The architecture patterns are the following:

- Direct Authentication
- Brokered Authentication

The design patterns are the following:

- Brokered Authentication: Kerberos
- Brokered Authentication: X.509 PKI
- Brokered Authentication: Security Token Service (STS)

Direct Authentication

Context

A client needs to access a Web service. The Web service requires the client to present credentials for authentication so that additional controls such as authorization and auditing can be implemented.

Problem

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

Forces

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

- **The credentials that the client presents to the Web service are based on shared secrets, such as passwords**. Authentication of individual users is often performed with passwords. Computers and applications often use higher quality secrets that are more secure than passwords. The client and the Web service must exchange the shared secrets securely before interaction is possible. The exchange of shared secrets must occur through an out-of-band mechanism.
- **The Web service can validate credentials from the client against an identity store**. The Web service must have direct access to the identity store, including appropriate permissions for accessing identity information.
- **The Web service is relatively simple, and does not require support for capabilities such as single-sign on (SSO) or support for non-repudiation**. In these circumstances an effective, low cost solution that does not use an authentication broker may be possible.
- **The client and the Web service trust one another to manage credentials securely**. In this situation, both parties should consider the credentials as equal in value to the information and services they protect. If either the Web service or the client manage the credentials in an insecure manner, neither party can be sure that the mishandled credentials prove the identity of the user or application.

Solution

Use direct authentication where the Web service acts as an authentication service to validate credentials from the client. The credentials, which include proof-of-possession that is based on shared secrets, are verified against an identity store.

Participants

Direct authentication 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.
- **Service**. The service is the Web service that requires authentication of a client prior to authorizing the client.
- **Identity store**. The entity that stores a client's credentials for a particular identity domain.

Process

Figure 1.4 identifies the tasks that occur during direct authentication.

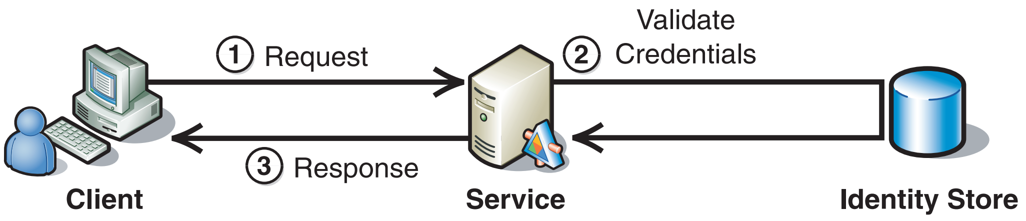

Figure 1.4
The direct authentication process

As illustrated in Figure 1.4, the following steps describe the direct authentication process:

1. The client sends a request to the Web service, attaching credentials to the request message.
2. The Web service validates the credentials against an identity store and makes authorization decisions about the client.
3. The Web service returns a response to the client. (This step is optional.)

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 Direct Authentication pattern include the following:

- It represents an uncomplicated model for authenticating clients without the need for an authentication broker.
- If the shared secret between a requester and service is compromised, only the relationship between those two parties is compromised and not the entire model.

Liabilities

The liabilities associated with the Direct Authentication pattern include the following:

- Direct authentication does not provide single sign on capabilities. Without single sign on, the client may be forced to authenticate prior to every Web service call or to cache the user's credentials within the application. If the user's credentials include a password, caching the password is not recommended because it may pose a security risk.
- The decentralized nature of direct authentication requires that the trust relationship be managed between each point in the communication, as shown in Figure 1.5.

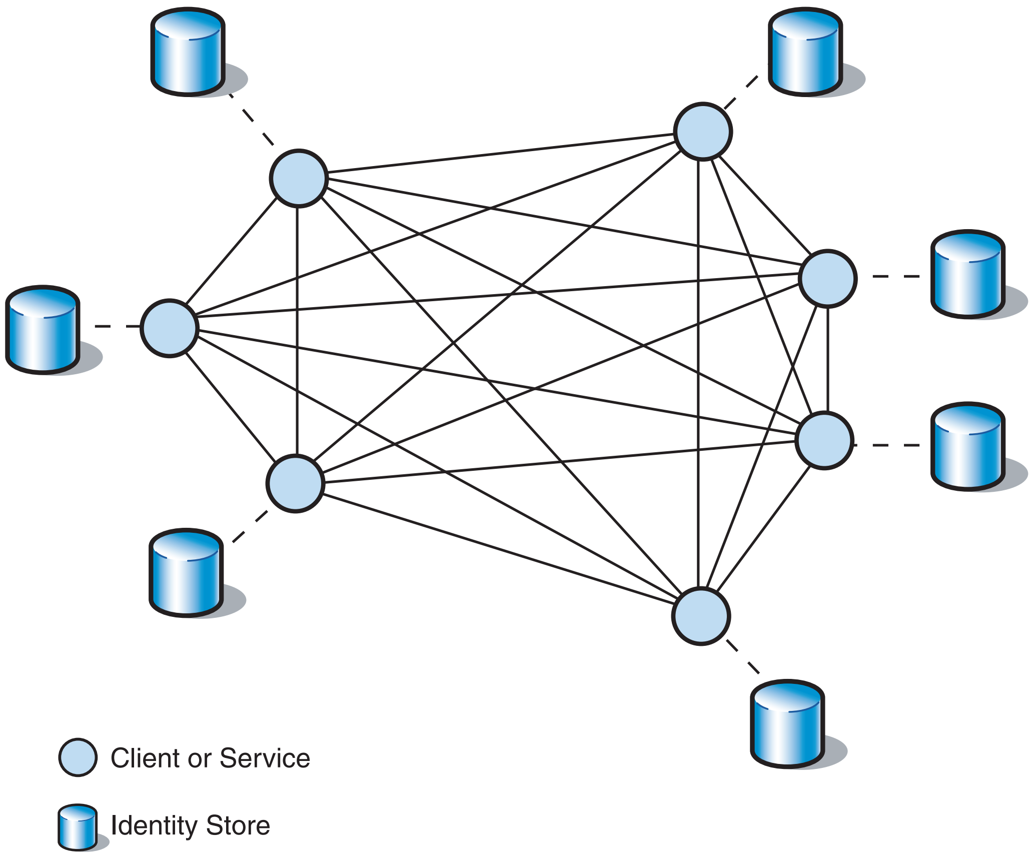

Figure 1.5
Trust relationships between points of communication in direct authentication

Each line in Figure 1.5 represents a discreet trust relationship established by authentication with a shared secret. As the number of discreet relationships between clients and services increases, each with potentially different identity stores, the challenges of managing and distributing secrets becomes more complicated.

- If a client calls a Web service frequently, the use of direct authentication can increase latency, because the Web service typically authenticates against a remote identity store.
- Data ownership and synchronization issues can occur if each of several services has its own identity store to authenticate the same client. This is because the client's credentials may need to be duplicated across multiple identity stores.

Security Considerations

Security considerations associated with the Direct Authentication pattern include the following:

- An attacker can impersonate the client if he or she intercepts the client's shared secret. The identity secret may be obtained if it is unprotected in transit or successfully guessed offline. You should use encryption to provide data confidentiality for this data. For more information, see Data Confidentiality in Chapter 2, "Message Protection Patterns."
- A shared secret is sensitive data and must be secured whenever it is persisted — even if it will be held for only a short time in a message queue. Shared secrets must be protected when stored in an identity store. If an attacker gains unauthorized access to an identity store that stores passwords in plaintext, all passwords in the identity store are immediately compromised. This allows the attacker to impersonate any user. Most authentication services such as Active Directory and Lightweight Directory Access Protocol (LDAP)-enabled directory services use identity stores that store passwords as either hashed, encrypted, or both. However, if you implement a custom identity store such as a database, you must ensure that the passwords are protected. Although a brute force or pre-computed dictionary attack is possible against a hashed password, hashing the passwords in the database will protect them from immediate disclosure in the event that an attacker gains access to them. For more information about how to hash passwords in a database, see Implementing Direct Authentication with UsernameToken in WSE 3.0 in Chapter 3, "Implementing Transport and Message Layer Security."
- If a client calls a Web service after a user has authenticated, it must cache the username and password locally for presentation on subsequent calls to the Web service for direct authentication. Caching secrets, such as passwords, increases the risk of disclosure if an attacker is able to gain access to the cache or flush the contents of the cache to an accessible location. You should secure the cache mechanism so that its confidentiality and integrity can be maintained to prevent disclosure or tampering. If the client is a Web application in a Web farm, you may want to consider brokered authentication instead of direct authentication. Otherwise, the user may be forced to re-authenticate if a request is routed to a different server in the farm after the user has authenticated.

Related Patterns

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

The following child patterns are related to the Data Authentication pattern:

- **Implementing Direct Authentication with UsernameToken in WSE 3.0**. This implementation pattern focuses on using direct authentication at the message layer.
- **Implementing Transport Layer Security Using HTTP Basic over HTTPS**. This reference provides information about implementing direct authentication using Internet Information Services (IIS) with X.509 certificates at the transport layer.

The following alternate pattern is related to the Direct Authentication pattern:

- **Brokered Authentication**. This pattern is an alternative to direct authentication that describes how to prove a client's identity to an authentication broker for issuance of a security token, and then use the issued security token to authenticate with a service.

The following pattern may use the Direct Authentication pattern:

- **Brokered Authentication**. This pattern may use variations of direct authentication to prove a client's identity to an authentication broker for issuance of a security token.

Brokered Authentication

Context

A client needs to access a Web service. The Web service requires the application to present credentials for authentication so that additional controls such as authorization and auditing can be implemented.

Problem

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