Security Considerations

Security considerations associated with the Perimeter Service Router pattern include the following:

- The perimeter service router is often the only point of entry to the internal network for external clients. This can make it a prime target for attackers. To guard against an attack, you must harden the platform on which the perimeter service router is deployed.
- Although the perimeter service router can provide an extra layer of security between external clients and internal Web services on a private network, you should still ensure that you design secure Web services on the internal network. You should also ensure that communications between the perimeter service router and internal Web services are secured.

Related Patterns

The following child pattern is related to the Perimeter Service Router pattern:

- **Implementing Perimeter Service Router in WSE 3.0**. This pattern provides steps and recommendations to implement a perimeter service router in WSE 3.0. It also discusses extensibility points in the **SoapHttpRouter** class in WSE 3.0 that you can use to address advanced scenarios, such as validation and dynamic routing

Implementing Perimeter Service Router in WSE 3.0

Context

You are exposing Web services deployed in a private network to external applications. Access to the Web services and resources in the private network is restricted to authenticated users. Any applications external to the private network must use a perimeter service router to access the Web services and resources deployed in the private network.

Objectives

The objectives of this pattern are to:

- Use a perimeter service router to provide an additional layer of security for services exposed to external clients.
- Allow the perimeter service router to route information to internal Web services based on a location contained within a configuration file.
- Demonstrate how to implement a perimeter service router using the WSE 3.0 **SoapHttpRouter** class.
- Discuss extensibility points in the **SoapHttpRouter** class in WSE 3.0 that you can use to address advanced scenarios, such as validation and dynamic routing.

Content

This pattern consists of the following sections:

- **Implementation Strategy**. This section provides a high-level description of the strategy to implement a perimeter service router.
- **Implementation Approach**. This section describes the steps required to implement this pattern:
  - General setup
  - Configure the external application
  - Configure the service router
  - Configure the service
- **Resulting Context**. This section outlines the benefits, liabilities, and the security considerations related to this pattern.
- **Extensions**. This section describes how to extend the base pattern to add more functionality for the router, including security policy enforcement.

Note: The code examples in this pattern are also available as executable QuickStarts on the Web Service Security community workspace.

Implementation Strategy

The implementation strategy for this pattern includes the following:

- Implement a perimeter service router in WSE 3.0, and deploy it as the service boundary between the perimeter network and the private network.
- Configure a handler to forward incoming requests from the router to the service.
- Create a routing referral cache that specifies the endpoint Uniform Resource Identifier (URI) of the service.

As an intermediary, the presence of the perimeter service router is not known to external applications. The service router represents the outward interface for the service that is deployed in the private network. External applications appear to communicate directly with the target service deployed to the internal network, although in reality they use an external URI for the service that is provided by the perimeter service router.

The Perimeter Service Router design pattern describes the perimeter service router as an intermediary that decouples internal services from external applications. This implementation pattern provides a more detailed description of that process.

Note: To fully understand this pattern, you must have some familiarity and experience with the .NET Framework, WSE 3.0, and Web service development.

Participants

Implementing a perimeter service router in WSE 3.0 involves the following participants:

- **External application**. An application outside the private network that needs to access the Web services in the private network.
- **Perimeter service router**. A Web service that provides access to Web services in the private network.
- **Service**. One or more Web services that the perimeter service router can access.

The following diagram displays the participants and their relation to each another in the private network, the perimeter network, and the public network.

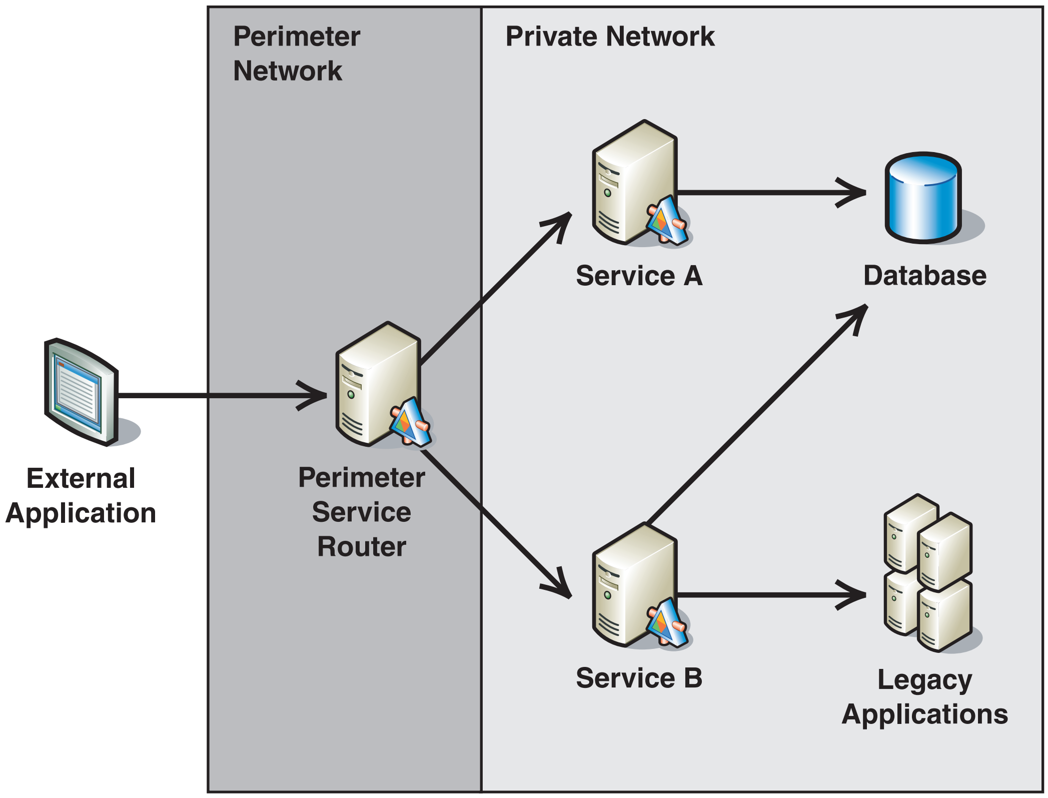

Figure 6.3
The deployment of a perimeter service router

Note: The code examples provided in this implementation pattern display the router and the service deployed on the same host for demonstration purposes only. Normally, you should deploy the perimeter service router to a server located on the perimeter network, and then deploy the service to a private network segment, as the previous diagram indicates.

Process

The Perimeter Service Router pattern provides a high-level overview of the perimeter service router functionality. This pattern describes the same process with refinements that are specific to this implementation. Figure 6.4 illustrates the functionality of the perimeter service router.

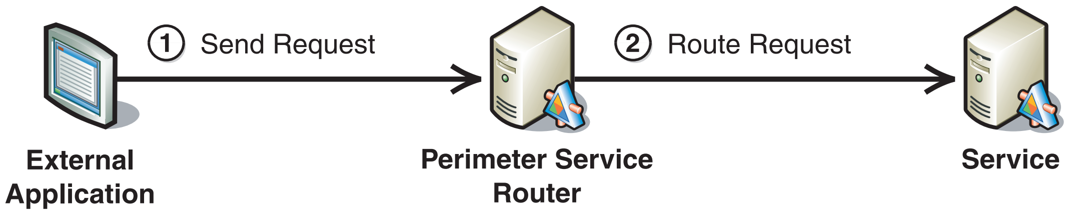

Figure 6.4
The functionality of a perimeter service router

The following steps show how the perimeter service router functions:

1. **The external application sends a request message**. The request message is addressed to the service's external interface as defined in the **<r:for>** entry in the referral cache on the perimeter service router.
2. **The perimeter service router forwards the request message to the service**. The perimeter service router directs the message to the target service URI defined in the **<r:go>** entry in its referral cache.

Implementation Approach

This section describes how to implement the pattern. The section is divided into four major tasks:

1. **General setup**. This task provides steps that apply to all applications for this pattern.
2. **Configure the external application**. This task lists the steps required to configure the external application to work with the perimeter service router.
3. **Configure the perimeter service router**. This task lists the steps required to configure policy and code on the perimeter service router.
4. **Configure the service**. This task lists the steps required to configure policy and code on the service to work with the perimeter service router.

This document describes the steps specific to implementing the perimeter service router. However, this document does not include details about how to implement authentication or message protection between the external application and service. For more information about authentication and securing communication between the external application and the service, see the following patterns:

- Direct Authentication in Chapter 1, "Authentication Patterns."
- Brokered Authentication in Chapter 1, "Authentication Patterns."
- Data Confidentiality in Chapter 2, "Message Protection Patterns."
- Data Origin Authentication in Chapter 2, "Message Protection Patterns."

Note: For the code examples included in this pattern, an ellipsis (...) is used where segments of code, such as class declarations and designer-generated code have been omitted. You must name variables, methods, and return values and ensure that they are of the appropriate type for the client application.

General Setup

You must install WSE 3.0 on the computers that you use to develop WSE 3.0-enabled applications. Once WSE 3.0 is installed, you must enable the perimeter service router and the service to support WSE 3.0. You can achieve this by performing the following steps:

► **To enable a Visual Studio 2005 project to support WSE 3.0**

1. In Visual Studio 2005, right-click the application project and select **WSE Settings 3.0**.
2. On the **General** tab, select the checkboxes for the following options:
   a. **Enable this project for Web Services Enhancements**.
   b. **Enable Microsoft Web Services Enhancement SOAP Protocol Factory**
3. Click **OK**.

Configure the External Application

The external application requires no special configuration to use the perimeter service router in order to communicate with resources on the private network. However, the Web service publisher must create a copy of the service's WSDL file and change the URI to the perimeter service router's URI for external clients. The external copy of the WSDL file should contain all of the Web service operations that the router publicly exposes. This is the WSDL that the external application would use to generate its proxy to communicate with the Web service.

Configure the Perimeter Service Router

To configure the perimeter service router, you need to create an entry for a SOAP router in the perimeter service router's configuration file, and then specify the location of a referral cache. You can achieve this by performing the following steps.

► **To configure the perimeter service router**

1. In Visual Studio 2005, right-click the service router project, and select **WSE Settings 3.0**.
2. On the **Routing** tab, click **Add**.
3. In the **Type** drop-down list box, type **Microsoft.Web.Services3.Messaging.SoapHttpRouter, Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35**. This sets the class that WSE 3.0 uses to process messages for routing.

   Note: At the time that this document was published, the default value that was available in the drop-down list box, **Microsoft.Web.Services3.Routing.RoutingHandler, Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35** would not function properly. You should therefore use the value specified in Step 3 above.

4. In the path box, type the name of an external interface for the Web service, or if the service router will handle routing for many Web services, type "***.asmx**."

   In this pattern, the service is exposed through the router as **ExternalService.asmx**. This is reflected in the URI that the external application uses as the address for its request messages. For example, if the router is deployed to **http://perimeterserver/router/**, the external URI that the external applications use to communicate with the service through the router is: **http://perimeterserver/router/ExternalService.asmx**.
5. In the **Verb** drop-down list box, select * to route messages based on all verbs, and then click **OK**.
6. In the **Referral Cache** box, type a name for the referral cache, such as **referralCache.config**. For security reasons, name the referral cache with a .config suffix. For more information, see the Security Considerations section in this pattern.
7. Create a new referral cache file, or copy and modify an existing referral cache file, and then add it to the perimeter service router project.

Note: The account that the perimeter service router runs under must have read and write permissions for the referral cache file.

The following is an example of a routing referral cache for a perimeter service router.

```
<?xml version="1.0" ?>
<!-- This is the referral cache file that forwards calls through the router to a
service -->
<r:referrals xmlns:r="http://schemas.xmlsoap.org/ws/2001/10/referral">
   <r:ref>
      <r:for>

<r:exact>http://localhost/PerimeterServiceRouter/Router/ExternalService.asmx</r:ex
act>
      </r:for>
      <r:if />
      <r:go>

<r:via>http://localhost/PerimeterServiceRouter/Service/InternalService.asmx</r:via>
      </r:go>
      <r:refId>uuid:093DC599-FD40-4bd3-B15F-02698D8EBFC2</r:refId>
   </r:ref>
</r:referrals>
```

The URI specified in the previous code sample for the **<r:for>** element is the URI for the perimeter service router. The **<r:go>** element contains the URI for the service to which requests are routed. The **<r: via>** element specifies a URI to reroute the SOAP message. When there are multiple **<r: via>** elements, the SOAP request is routed only to the first **<r: via>** element. For more information on referral cache syntax, see How to: Configure the WSE SOAP Router.

Once the routing handler is configured, you can see an entry for an HTTP handler in the perimeter service router's configuration file that should look like the information in the following code sample.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
 ...
 <system.web>
  <httpHandlers>
<add type=" Microsoft.Web.Services3.Messaging.SoapHttpRouter,
Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" path="ExternalService.asmx" verb="*" />
    ...
  </httpHandlers>
  ...
  </system.web>
  ...
</configuration>
```

Configure the Service

Web services that are not WSE-3.0 enabled will normally accept routed messages from a perimeter service router. However, a WSE 3.0-enabled Web service will reject a message that is not specifically addressed to it, unless it is configured to accept a message addressed to a different party. Because external applications use the external URI for the service on the perimeter service router to address request messages, you must configure a WSE-3.0 enabled service to accept messages that are addressed to the perimeter service router. You can do this by adding a **SoapActor** attribute above the class declaration code for your Web service class, as defined in the following code sample.

```
...
using Microsoft.Web.Services3.Messaging;
...
[SoapActor("http://localhost/PerimeterServiceRouter/Router/ExternalService.asmx")]
```

Substitute the URI in this code sample for the one that you use to externally expose your service on the perimeter service router.

Resulting Context

This section describes some of the more significant benefits, liabilities, and security considerations of using this implementation 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 Implementing Perimeter Service Router in WSE 3.0 pattern include the following:

- You can use the perimeter service router to extend the service boundary to the perimeter of the private network, which allows you to consolidate common perimeter security functions on the perimeter service router. For more information about this topic, see the Extensions section.
- You can take servers that host internal Web services offline for maintenance without affecting the external interface. You can accomplish this by configuring the perimeter service router to route messages to an alternate server while the staff performs maintenance on the primary server.

Liabilities

The liabilities associated with the Implementing Perimeter Service Router in WSE 3.0 pattern include the following:

- Each additional service interface that a service provides increases the amount of work required to change the functionality exposed by the perimeter service router.
- The implementation may add complexity and performance overhead that may not be justified for simple service-oriented applications.
- Internet Information Services (IIS) 6.0 locks the routing referral cache file for a deployed perimeter service router, which prevents direct modification of the cache. If you attempt to make modifications directly to the referral cache, you must first restart IIS 6.0 before you can save them. This requirement may affect the availability of the service router or other Web applications that the server may host. To resolve this issue, you can create another referral cache file, and then update the perimeter service router's Web.config file to point to the new referral cache file.

Security Considerations

Security considerations associated with the Implementing Perimeter Service Router in WSE 3.0 pattern include the following:

- External interfaces such as perimeter service routers are typically prime targets for attackers that represent major entry points to the private network.
- You should name your referral cache file with a .config extension. IIS 6.0 filtering prevents clients from accessing the contents in .config files. If you name the referral cache with a different extension, the filtering may not work properly and a client could access the contents to expose the internal URI of the Web service.

Extensions

This section discusses an extension that you can use to increase the functionality of the perimeter service router.

Extension 1 — Using the Perimeter Service Router as a Policy Enforcer

You can extend the perimeter service router to perform additional security functions by implementing a custom service router class that extends the **SoapHttpRouter** class. The custom service router can act as a policy enforcer to authenticate clients, perform message validation, reject replayed messages, attribute activity to a specific user or organization, and perform other security functions.

To use a custom service router class, the routing handler you create (that is described in the Implementation Approach section) must implement a custom router class that you can derive from the default **SoapHttpRouter** class.

You can override the following methods in the custom service router:

- **GetRequestPolicy**. This method allows you to implement logic that dynamically determines which policy is enforced for any request message received from the external application. This capability is useful for implementing message validation, rejecting replayed messages, and authenticating the caller.
- **GetForwardRequestPolicy**. This method allows you to implement logic that dynamically determines which policy is enforced for request messages passed from the perimeter service router to the service. This capability is useful for specifying policies for the perimeter service router to sign and encrypt an incoming request message from the external application.
- **ProcessRequestMessage**. This method allows you to implement logic that dynamically routes an incoming request message based on message content or other factors. For example, you can use this method to route an incoming message to an alternate destination while the primary recipient is offline and unable to accept request messages. When you override the **ProcessRequestMessage** method, the service might not use a referral cache unless it calls the **ProcessRequestMessage** method of the parent **SoapHttpRouter** class.

You can deploy the perimeter service router as an entry point for a trusted subsystem, which means that the service authenticates the routed request message, based on the perimeter service router's credentials instead of the original caller's.

In a trusted subsystem model, if you need to forward security claims from the original caller in the routed message, you must create a custom filter to add the claims to the security header of the request message. For more information about trusted subsystems, see Trusted Subsystem in Chapter 4, "Resource Access Patterns."

If you need to retain the security context of the original caller while doing anything other than simple pass-through routing as described in the base pattern, the external application must add claims to the request message to satisfy policy requirements for the internal service. In this case, the perimeter service router adds its own claims to the request message in addition to those that the requestor added. It then forwards the message to the internal service. The internal service then processes claims on request for both the external application, as the originating client, and the router to provide assurance that the message has passed through the router. For details on this approach, see the "SecureRoutingToUltimateReceiver" QuickStart sample in the WSE 3.0 QuickStarts folder.

For more information about implementing SOAP routers in WSE 3.0, see: Routing SOAP Messages with WSE.

More Information

"Service Interface Pattern" in *Enterprise Solution Patterns Using Microsoft .NET* on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpatterns/html/DesServiceInterface.asp*.

For more information about using the WseWsdl3.exe utility, see the "WSDL to Proxy Class Tool" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/fbefe453-3851-439b-9c10-fb036b59ff81.asp*.

For more information on referral cache syntax, see "How to: Configure the WSE SOAP Router" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/6414f229-cead-48af-a293-cb893c24c0e6.asp*.

For more information about implementing SOAP routers in WSE 3.0, see: "Routing SOAP Messages with WSE" on MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/b41230fb-d0e1-48b1-88c0-3daf7a40c9e8.asp*.