Using a reverse proxy server with WebSphere Commerce

Shash Anand
Scott Guminy

December 07, 2005

This article describes how to configure the IBM® HTTP Server as a reverse proxy server for WebSphere® Commerce. A reverse proxy server provides an additional layer of security, protects HTTP servers further up the chain, and improves the performance of Secure Sockets Layer (SSL) requests.

Introduction

A reverse proxy server adds an additional layer of protection between your WebSphere Commerce server and the Internet. This article provides detailed instructions on how to configure the IBM HTTP Server as a reverse proxy server for WebSphere Commerce, and provides a sample configuration file. This article also describes how WebSphere Application Server maps HTTP requests to application servers. This is useful when troubleshooting your reverse proxy server configuration.

Reverse proxy overview

A reverse proxy server is a special HTTP server that prevents direct access to the content HTTP server. All requests for the content HTTP server go through the reverse proxy server. The process is illustrated in Figure 1.

Figure 1. Communication using a reverse proxy server

As shown in Figure 1, the browser makes an HTTP request to what it thinks is the content HTTP server, but it is actually the reverse proxy server. The reverse proxy server forwards the request to
the content HTTP server. The content HTTP server sends the response back to the reverse proxy server, and the response is sent back to the browser. You can put firewalls in place to increase security. In the end, the browser does not know that the request was re-routed to an alternative server. It thinks it is accessing the content HTTP server directly.

**Difference between forward and reverse proxies**

A forward proxy is typically used when you have internal users of your network trying to access external sites. The internal users must configure the client browser to use the proxy to access external sites.

In contrast, a reverse proxy server is typically used to appear as the actual Web server to the client. The reverse proxy server takes all requests, and sends the requests to the content HTTP server. When the proxy server receives a response, it sends the results back to the client as if it were the original server. A reverse proxy is typically used when you want clients to access a server that is behind a firewall.

**Advantages of using a reverse proxy server**

The two main advantages of using a reverse proxy server are system security and performance:

- **Security**: The reverse proxy server provides an additional layer of security and can protect HTTP servers further up the chain. If you are using a firewall between the reverse proxy server and the content HTTP server, you can configure the firewall to allow only HTTP requests from the reverse proxy server.
- **Performance**: A reverse proxy server can increase the performance of your WebSphere Commerce server in several ways.
  - **Encryption/SSL acceleration**: You can equip the reverse proxy server with SSL acceleration hardware that can improve the performance of SSL requests.
  - **Caching**: The reverse proxy server can cache static content to provide better performance.
  - **Load balancing**: The reverse proxy server can balance the workload among several content HTTP servers. For example, you can split the static content between two content HTTP servers.

**Standard WebSphere Commerce HTTP Server configuration**

You use WebSphere Commerce Configuration Manager to create a WebSphere Commerce instance and configure HTTP access to the application. This configuration is done in the HTTP Server and in WebSphere Application Server. Each Web application archive (WAR) has its own configuration as described in the following list. `<long host>` is your fully qualified host name. `<short host>` is the first part of your host name. For example, commerce.mydomain.com. `<long host>` is your fully qualified host name. For example, commerce.

Stores.war:
Using a reverse proxy server with WebSphere Commerce

URI: /webapp/wcs/stores/*
WebSphere Application Server Virtual host
Name: VH_<instance>
Host Alias: <long host>:80
Host Alias: <short host>:80
Host Alias: <long host>:443 used for HTTPS access
Host Alias: <short host>:443
IBM HTTP Server (httpd.conf)
VirtualHost: <long host>:80
Alias /wcsstore
Alias /wcs
VirtualHost: <long host>:443
Alias /wcsstore
Alias /wcs
SSLEnabled

CommerceAccelerator.war:

URI: /webapp/wcs/tools/*
WebSphere Application Server Virtual host
Name: VH_<instance>_Tools
Host Alias: <long host>:8000 used for HTTPS access
Host Alias: <short host>:8000 used for HTTPS access
IBM HTTP Server (httpd.conf)
VirtualHost: <long host>:8000
Alias /adminconsole
Alias /wcsstore
Alias /accelerator
Alias /wcs
Alias /wcadmin
Alias /wcorgadmin
Alias /orgadminconsole
SSLEnabled

SiteAdministration.war

URI: /webapp/wcs/orgadmin/*
WebSphere Application Server Virtual host
Name: VH_<instance>_Admin
Host Alias: <long host>:8002 used for HTTPS access
Host Alias: <short host>:8002 used for HTTPS access
IBM HTTP Server (httpd.conf)
VirtualHost: <long host>:8002
Alias /adminconsole
Alias /wcsstore
Alias /accelerator
Alias /wcs
Alias /wcadmin
Alias /wcorgadmin
Alias /orgadminconsole
SSLEnabled

OrganizationAdministration.war:
**Processing an HTTP request with Web server plug-in**

The Web server plug-in installed in IBM HTTP Server is used to route a request to a particular application server. WebSphere Application Server generates a configuration file, plugin-cfg.xml, that is used by the IBM HTTP Server plug-in to determine how to route requests to WebSphere Application Servers. The plugin-cfg.xml file contains the mappings of virtual hosts and URIs to the host names and port numbers of the application servers that run the application identified by that virtual host and URI.

Here is an example of how the plug-in determines to which application server the request is sent:

2. IBM HTTP Server receives the request and hands it over to the WebSphere Application Server plug-in.
3. The plug-in determines if it should handle the request by performing the following steps:
   1. The URL is broken up into two parts: Host (www.mycompany.com:80) and URI (/webapp/wcs/stores/MyStore/index.jsp).
   2. The plug-in finds the virtual host defined in the plugin-cfg.xml file that best matches the request. Because virtual hosts can have wildcards in them, an exact match would take priority. For example, take two virtual hosts with the following configurations:
      - default_host - Host Alias: *:80
      - WC_demo - Host Alias: www.mycompany.com:80
   
      In this case, WC_demo would take priority because it is an exact match with the host name of the HTTP request.
3. Once the virtual host is determined, the plug-in will find all URIs that are mapped to that virtual host. The URI to VirtualHost Mapping is configured when installing EAR files into WebSphere Application Server. In this example, the URI of the WebSphere Commerce Stores.war application would match the incoming request (/webapp/wcs/stores/*).
4. If a URI match is found, the plug-in determines which server or server cluster is configured to run the application identified by this virtual host and URI combination. This mapping is set when an application is installed. The following events take place:
   1. The plug-in sends an HTTP request to the WebSphere Application Server to process the request.
2. The response is sent back to the browser.
3. If a server is not found, IBM HTTP Server will process the request using its httpd.conf configuration file.

**Configuring IBM HTTP Server as a reverse proxy server for WebSphere Commerce**

Before you configure the HTTP Server, complete the following tasks:

1. Install WebSphere Commerce.
2. Create a WebSphere Commerce instance.
3. Publish a store on your WebSphere Commerce instance.
4. Ensure that you can access the WebSphere Commerce instance via the Web server configured by WebSphere Commerce.

**Desired topology**

Figure 2 shows the topology that you will configure in this example.

**Figure 2. Sample WebSphere Commerce topology with a reverse proxy server**

You will introduce a reverse proxy server to protect the Web server installed on the application tier. You will configure the reverse proxy server to forward HTTP requests to the Web server on the application tier that was configured by WebSphere Commerce. The following ports are used by WebSphere Commerce: 80, 443, 8000, 8002 and 8004. The minimum set of ports required to be exposed by the proxy server are 80 and 443. These ports are used by the Stores Web Module and give external access to your published stores. If you are using a Business to Business store model, you may also want to expose the ports for the other Web modules. In this example, you will expose all WebSphere Commerce ports on the reverse proxy server.

For this configuration task, assume the content Web server is content.mycompany.com. Without the proxy server, you would access the WebSphere Commerce server via the following two addresses:

- [http://content.mycompany.com/webapp/wcs/stores/MyStore/index.jsp](http://content.mycompany.com/webapp/wcs/stores/MyStore/index.jsp)
- [https://content.mycompany.com:8000/accelerator](https://content.mycompany.com:8000/accelerator)

You will introduce a reverse proxy server that will handle requests for the internet. This system responds to the commerce.mycompany.com host name. Hence, the following URLs access the site:

- [https://commerce.mycompany.com:8000/accelerator](https://commerce.mycompany.com:8000/accelerator)

Using a reverse proxy server with WebSphere Commerce
• http://commerce.mycompany.com/webapp/wcs/stores/MyStore/index.jsp, Or
• https://commerce.mycompany.com:8000/accelerator

Installing IBM HTTP Server on reverse proxy machine

The reverse proxy server software used in this example is IBM HTTP Server 2.0.47.1. Note that there are important fixes in this version of IBM HTTP Server that are required when using the proxy function. Previous versions may not work.

Follow the installation instructions to install IBM HTTP Server. The following example assumes you have installed IBM HTTP Server into the c:\IHS2.0.47.1 directory.

Configuring IBM HTTP Server 2.0.47.1

1. Open the following file in a text editor, c:\IHS2.0.47.1\conf\httpd.conf.
2. Enable the Proxy Modules by removing the # character from (uncommenting) the following lines:
   
   #LoadModule proxy_module modules/mod_proxy.so
   #LoadModule proxy_http_module modules/mod_proxy_http.so

3. Enable the reverse proxy server to listen on ports 80, 443, 8000, 8002, and 8004 by adding one Listen directive for each port as follows:
   
   Listen 0.0.0.0:80
   Listen 0.0.0.0:443
   Listen 0.0.0.0:8000
   Listen 0.0.0.0:8002
   Listen 0.0.0.0:8004

4. Enable the SSL Module by uncommenting the following line:
   
   #LoadModule ibm_ssl_module modules/mod_ibm_ssl.so

5. Virtual host directives (<Virtual Host>) are used to configure specific properties for a particular virtual host. Each virtual host refers to a different IP address, different port number, or a different host name for the server. To add virtual hosts, complete the following steps:

   1. Within each virtual host, you must add the ProxyPass directive. This directive maps the local URL to a proxy request to the content HTTP server, which is specified in the second parameter. For example, ProxyPass / http://content.mycompany.com/ will forward all requests made to the proxy server to http://content.mycompany.com/.

   2. Within each virtual host, you must also set the ProxyPreserveHost directive to "on". This line informs the proxy server to set the host name on the proxy request to that of the proxy server. In this example, the proxy host name is commerce.mycompany.com and the content host name is content.mycompany.com. Enabling this directive ensures that when an HTTP request is sent to the content.mycompany.com host, the original host name (commerce.mycompany.com) is set in the host header on the HTTP request. This line is important because WebSphere Commerce performs redirects, and the host name specified in the redirect should be the proxy, not the content HTTP server.

   3. If this virtual host is SSL enabled, you must also include the SSLProxyEngine on directive. Set your SSL key file that contains your SSL certificate signed by a certificate authority (CA) using the following KeyFile directive:

   KeyFile c:\IHS2.0.47.1\ssl\mykey.kdb

4. If your content HTTP server's SSL certificate is self-signed, ensure that the key file previously specified includes the certificate of the CA that signed your self-signed
certificate. If you do not, the reverse proxy server is not able to forward requests to the content HTTP server because it does not trust the CA that signed the content HTTP server's SSL certificate. The simplest way to do this is to use the key file from your content server's configuration.

5. You must add one VirtualHost directive for each port of the Commerce server. The following example is for port 443 (SSL):

```apache
<VirtualHost commerce.mycompany.com:443>
  Keyfile "c:\IHS2.0.47.1\ssl\keyfile.kdb"
  SSLEnable
  SSLClientAuth 0
  ProxyPass / https://content.mycompany.com/
  ProxyPreserveHost on
  SSLProxyEngine on
</VirtualHost>
```

You can use the complete `httpd.conf` configuration file provided in this article.

6. Save the configuration file.

**Configuring WebSphere Commerce to respond to new host name**

Because the reverse proxy server has a different host name, you must update WebSphere Application Server so that the plug-in can correctly map the inbound request to your application server.

You can do this by adding the new host name to each virtual host that WebSphere Commerce is configured to use. You must update the following virtual hosts:

- VH_<instance>
- VH_<instance>_Tools
- VH_<instance>_Admin
- VH_<instance>_OrgAdmin

Complete the following steps to update each virtual host:

1. Open the WebSphere Administrative Console.
2. In the navigate panel, expand Environment.
3. Click on Virtual Hosts.
4. Click on the virtual host you are updating.
5. Click on Host Aliases. A list of existing aliases is displayed. Take note of the existing port number used for the host aliases. You will use this for the next step.
6. To create a new host alias, click New.
7. In the Host name field, enter `commerce.mycompany.com`. This is the host name of the reverse proxy server.
8. Enter the port number into the port field.
9. Click OK.
10. Click OK.
11. For each virtual host, repeat Steps 4-10.
12. Save the configuration.
13. Regenerate your plug-in configuration file by following the instructions in the WebSphere Commerce Installation Guide.
14. Restart your WebSphere Commerce HTTP server.
15. Restart your WebSphere Commerce server.

Verifying new configuration of the proxy server

At this point, you have completed the configuration of the proxy server and have updated the WebSphere Commerce Server to recognize the new host name. To verify the configuration, complete the following steps:

1. Start the reverse proxy server.
2. Open a Web browser and navigate to the following URL, http://commerce.mycompany.com/webapp/wcs/stores/servlet/?DEBUG=1. The following output is displayed:

   Debug mode
   QueryString=DEBUG=1
   Method=GET
   ContentLength=-1
   ServerName=commerce.mycompany.com
   RequestURI=/webapp/wcs/stores/servlet/
   SERVLET_PATH=/servlet
   PathInfo=/
   Command=

3. Try the same URL, but with SSL: https://commerce.mycompany.com/webapp/wcs/stores/servlet/?DEBUG=1. The following output is displayed:

   Debug mode
   QueryString=DEBUG=1
   Method=GET
   ContentLength=-1
   ServerName=commerce.mycompany.com
   RequestURI=/webapp/wcs/stores/servlet/
   SERVLET_PATH=/servlet
   PathInfo=/
   Command=

If either of these fails, you should enable plug-in tracing to determine if the WebSphere Application Server plug-in is sending the request to your WebSphere Commerce Application Server. For information on tracing the WebSphere Application Server plug-in, see the WebSphere Application Server Information Center.

If these debug URLs work, try displaying the store that you have published using the proxy server's host name.

Conclusion

This article introduced a reverse proxy server to increase the security of your WebSphere Commerce server. It provided detailed instructions on how to configure IBM HTTP Server as a reverse proxy server for IBM WebSphere Commerce. By configuring a reverse proxy, you added
an extra layer of protection between the Internet and your WebSphere Commerce server. In addition, you can take advantage of the performance enhancements that a reverse proxy server provides.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code sample</td>
<td>httpd.conf.zip (HTTP</td>
<td>FTP</td>
</tr>
</tbody>
</table>
Related topics

- See the Apache Software Foundation site for more information about Apache.
- Learn about the WebSphere Application Server product in the WebSphere Application Server Information Center.
- Learn about the WebSphere Commerce product in the WebSphere Commerce Information Center.
- Visit the developerWorks WebSphere Commerce zone for technical and how-to information.