WebSEAL-Lite 1.0

User's Guide

June 20, 2001

Table of Contents

1 Introducing WebSEAL-Lite

1.1 Product Overview

Generally, Web servers that provide resources to authenticated users, maintain and enforce the authentication and authorization of these users. However, setting up independent authentication points within a secured domain makes the administration of these user databases unwieldy. Also, multiple authentication points within a domain may require the end-user to maintain multiple user IDs and passwords, and be prompted to authenticate multiple times within the domain.

WebSEAL-Lite is an authentication and authorization plugin for *Web Traffic Express* which provides a central authentication point for all users who wish to access resources within a secured domain. It combines the caching Web proxy of Web Traffic Express with the authorization engine of *Policy Director* to deliver protected resources to authorized users.

1.2 Required Software

The proper software packages must be installed before installing WebSEAL-Lite. If a required software package is not found, you will not be able to install WebSEAL-Lite. The following software packages are required to install WebSEAL-Lite:

Machine Running WebSEAL-Lite

- Policy Director RTE 3.7.1
- LDAP Client 3.2
- Java Run Time Environment (JRE) 1.3
- Web Traffic Express 3.6
- WebSEAL-Lite 1.0

Machine Running Policy Director

- DCE Client & Server 3.1
- Policy Director Server 3.7.1
- LDAP Client & Server 3.2

2 Installing WebSEAL-Lite

2.1 Installation Procedure

WebSEAL-Lite is installed using the operating system standard software installation tool. Before installing WebSEAL-Lite, verify that the required software packages listed in the previous section have already been installed. Then, follow the steps below to install WebSEAL-Lite:

- **Download WebSEAL-Lite**
- **Install WebSEAL-Lite**

Download WebSEAL-Lite

Download WebSEAL-Lite into a temporary directory from the following FTP site:

ftp://wsluser:wsluser@swift.raleigh.tivoli.com/Current

Download the corresponding package for your operating system. Also, be sure to download the latest ***libpdauthzn*** library from the ***PolicyDirector*** directory. After downloading this library, copy it over the corresponding installed Policy Director library on your file system. If you do not intend to install Policy Director because you are a *WebSphere Everyplace Suite* user, be sure to download the stub version of ***libpdauthzn***, so that WebSEAL-Lite will load successfully. The stub library should be placed in the ***/lib*** directory.

Install WebSEAL-Lite

Install WebSEAL-Lite using the installation tool for your operating system. The following installation tools are listed with their corresponding operating systems:

AIX ***installp*** or ***smit***

SUN ***pkgadd***

Linux ***rpm***

NT ***setup.exe***

After the software package has been installed successfully, it will need to configured before it can be used. The next section will explain that procedure.

3 Configuring WebSEAL-Lite

3.1 Configuration Procedure

WebSEAL-Lite is configured using the provided configuration tool, ***wslconfig.sh*** *(****wslconfig.exe on NT****)*. The configuration tool will perform the following key tasks:

- Create a WebSEAL-Lite Application User in Policy Director
- Create the Object Space for Web Traffic Express in Policy Director
- Create the LTPA Cookie Key File
- Configure Web Traffic Express to Load WebSEAL-Lite

After installing the software package, run the following command at the shell without any parameters:

```
# wslconfig.sh
```

You will be asked a series of questions regarding your Policy Director configuration. Some of the information you will need to configure WebSEAL-Lite will include:

- Policy Director Administrator user ID and password
- Policy Director LDAP configuration

If the configuration fails, it will automatically unconfigure itself. If you terminate the script before it completes, then be sure to unconfigure WebSEAL-Lite by issuing the command below, before reconfiguring it again:

```
# wslconfig.sh -u
```

After the configuration completes successfully, Web Traffic Express should be running with WebSEAL-Lite loaded as a plugin. A tool is provided in case you want to start it manually. The command, ***wslstartwte***, will start Web Traffic Express along with WebSEAL-Lite:

wslstartwte

If you want to automatically start Web Traffic Express whenever your system is rebooted, be sure to invoke this utility in your start up scripts. To stop Web Traffic Express, search memory to kill the process, ***ibmproxy***. You will also need to restart Web Traffic Express whenever you change the WebSEAL-Lite configuration files for the changes to become effective.

The configuration files are placed in the **etc/** sub-directory when WebSEAL-Lite is installed, and may be manually modified after the configuration tool completes successfully. There are three configuration files used by WebSEAL-Lite:

- Base Configuration File (*ibmwesas.conf*)
- Object Space Configuration File (*osdef.conf*)
- User Mapping Configuration File (*usermap.conf*)

The base configuration file contains initialization information, and points to the locations of the additional two configuration files. The configuration tool initializes this file with the information you supply it when you run it. The object space configuration file contains information WebSEAL-Lite uses to authorize users, and the user mapping configuration file is used to map single sign-on and certificate users to Policy Director users.

WebSEAL-Lite logs events and errors to the Web Traffic Express log files. To observe events as they occur, view these log files while WebSEAL-Lite is running.

After WebSEAL-Lite has successfully been configured, you may issue requests from your Web browser to Web Traffic Express using it either as a proxy, or by specifying the domain name of the Web Traffic Express machine in your URL as shown below:

```
http://<domain name of Web Traffic Express>
```

However, since no ACLs will have been enabled in the Policy Director object space, you will not be allowed to retrieve this request. Some familiarity with Policy Director will be required to administer access control using WebSEAL-Lite. The next section will address this and other related topics.

4 Understanding WebSEAL-Lite

4.1 Policy Director Concepts

Before you can exploit WebSEAL-Lite functionality, you will need to understand how Policy Director implements access control. Access control is achieved by integrating the following components of access control:

- **Users and Groups**
- **Access Control Lists (ACLs)**
- **Object Space**

Users and Groups

Users and Groups represent the individuals that the access control will be applied to. This is the first step in providing access control. Please refer to your Policy Director manual for instructions on how to create users and groups in Policy Director.

Access Control Lists (ACLs)

An *Access Control List (ACL)* is a list of users and groups along with their associated permissions. An ACL says nothing about what object a users permissions are being applied to. It simply states that a collection of users and groups have the specified respective permissions. Users and groups are not bound to one ACL. A user or group may appear in one or more ACLs. This is the second step in providing access control. Please refer to your Policy Director manual for instructions on how to create ACLs in Policy Director.

Object Space

The final piece of information needed to provide access control, is the object which the user will be accessing. Policy Director represents objects using a hierarchical representation called the *Object Space*, similar to the way an operating system represents files and directories on a file system. Access control is achieved by placing ACLs at the appropriate locations in the object space. The interpretation of each object in the object space along with the associated ACL is subject to each application. Policy Director simply tells the application what ACLs are associated with a given object in the object space.

Manually creating the object space can be an enormous task. Accordingly, WebSEAL-Lite provides a command line utility, ***wesosm*** *(Object Space Manager)*, to create and manage the object space. This utility will create and maintain the object space using the WebSEAL-Lite configuration files. The next sections will describe this utility in detail.

4.2 WebSEAL-Lite Concepts

Typically, when a user issues a request to a Web site using a browser, the object represented in the URL corresponds to an object on a Web server. WebSEAL-Lite provides access control by verifying that the user is allowed to access the requested object on the Web server, before allowing the request to complete. WebSEAL-Lite provides access control in the following modes of operation:

- **Reverse Proxy**
- **Forward Proxy**

Reverse Proxy

Since Web site content may span multiple Web servers for performance and content distribution, Web Traffic Express may be used as a reverse proxy to the internal or backend Web servers. This is accomplished by configuring the Web site public domain name on the Web Traffic Express machine, and specifying a route to the corresponding backend Web server, as illustrated in Figure 1.

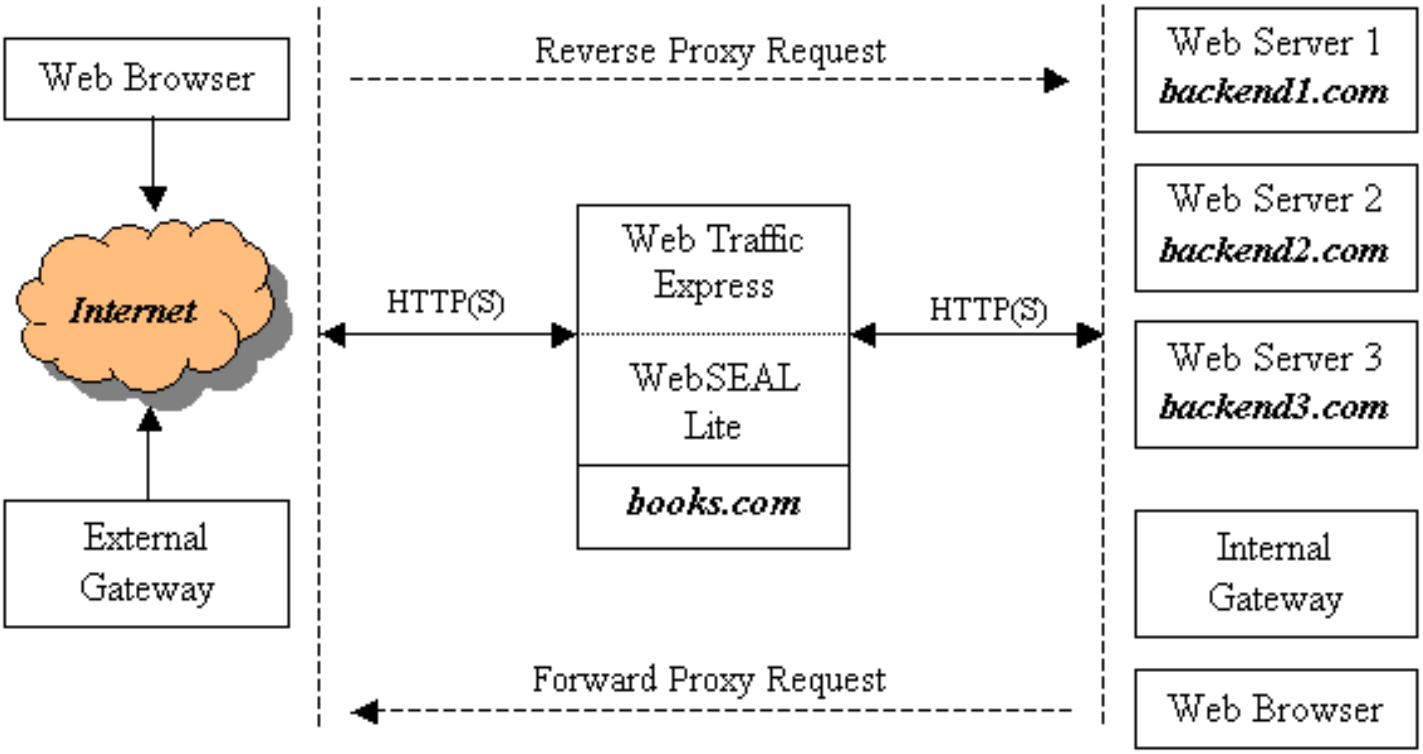

Figure 1 Reverse and Forward Proxy Requests

In this example, WebSEAL-Lite would be configured to provide access control to the objects on **books.com**. After a user had been successfully authorized, the request would be routed to the corresponding backend server either by WebSEAL-Lite, or though the use of a load balancing module, like the *Network Dispatcher Content Based Routing* module. WebSEAL-Lite performs simple URL mapping, similar to the functionality provided by the *Proxy* statements in the Web Traffic Express configuration file.

WebSEAL-Lite access control is configured through its object space configuration file, (***osdef.conf***). In this file, you would add the following entry to configure this Web site:

```
[Remote: /WebSEAL-Lite/reverse/books.com]

domains = books.com www.books.com

login_method = forms

form_login_file = http://books.com/pub/login.html

form_login_errorfile = http://books.com/pub/loginerr.html

form_logout_file = /account/logout.html

route = https://backend1.com
```

This entry tells WebSEAL-Lite to authorize all requests for **books.com** and **www.books.com** using **/WebSEAL-Lite/reverse/books.com** in the Policy Director object space, to use forms as the login method, and to map every URL to the same Web server. By default, WebSEAL-Lite checks **/WebSEAL-Lite/reverse/<domain name>** for reverse proxy requests. There are other options that you could associate with this server definition. If you do not explicitly specify a setting for an option, the setting for that option is inherited from the **[Global]** section of the configuration file.

Although ***forms*** was choosen as the login method in this example, there are other settings that may be

choosen for **login_method**. For example, ***basic*** may be selected for basic authentication and ***certificate*** may be selected for client certificate authentication. All of these settings, as well as options specific to each setting are described in more detail in the sample configuration file, `osdef.conf`.

Forward Proxy

WebSEAL-Lite may also be used to provide access control to outbound requests, as illustrated in Figure 2. Users on the internal side of the firewall could configure their browsers to use the same instance of Web Traffic Express as proxy to get to the Internet. By default, WebSEAL-Lite checks **/WebSEAL-Lite/forward/<domain name>** for forward proxy requests. But you can explicitly override this by creating a server definition in the object space configuration file as shown below:

```
[Remote: /WebSEAL-Lite/forward/blockedsites]

domains = games.com *.games.com *.competitor.com
```

In this example, all browser requests matching the above domain names would be redirected to the company browsing policy Web page. Alternatively, you could place an ACL at this location in the object space that would prevent anyone from going to the listed Web sites.

4.3 Creating the Object Space

In order for you to specify what objects users are allowed to access on a Web server, ACLs must be associated with the objects provided by the Web server. This means that the Web server object hierarchy needs to be represented in the Policy Director object space. The simplest way to do this, is to import the Web server file system into the Policy Director object space. After this has been done, ACLs may be placed on the desired objects. The WebSEAL-Lite Object Space Manager, ***wesosm***, generates the object space for the following Web servers:

- **Web Traffic Express**
- **Other Web Servers**

Web Traffic Express

Although Web Traffic Express is a proxy, it can function like a Web server when requests are made directly to the primary domain name of the Web Traffic Express machine. Typically, informational and error messages are stored in the proxy Web space. WebSEAL-Lite will enforce access control to these objects managed by Web Traffic Express. The following server definition in the configuration file is used to represent Web Traffic Express:

```
[Local: /WebSEAL-Lite/bookproxy.com]

domains = bookproxy.com

query_command = http://bookproxy.com/cgi-bin/query_contents?dirlist=/
```

The configuration tool executes ***wesosm*** to generate the object space for Web Traffic Express as its final step in configuring WebSEAL-Lite. After running the configuration tool, you should be able to place ACLs at the appropriate locations in the object space for Web Traffic Express. For example, informational and error pages should not require authorization for users to access.

Other Web Servers

The Object Space Manager, ***wesosm***, may also be used to query a remote Web server file system in order to create corresponding entries in the Policy Director object space. To do this, it reads the object space configuration file and creates object entries for each server definition in the file. In order to import a Web server file system using this utility, another supplied utility, ***query_contents***, must be placed in the ***cgi-bin*** directory on the Web server that is to be queried. Some configuration will be required to specify the root location of the Web server files. Please refer to Policy Director's WebSEAL Administration Guide (GC32-0684-00), for

information on how to configure the ***query_contents*** utility.

Afterward, an entry must be added to the object space configuration file, telling ***wesosm*** how to query the remote Web server file system. Using the example from the previous section, you would place the following entry in the configuration file:

```
[Remote: /WebSEAL-Lite/reverse/books.com]

domains = books.com www.books.com

query_command = http://backend1.com/cgi-bin/query_contents?dirlist=/
```

After the entry has been added to the configuration file, run the Object Space Manager from the WebSEAL-Lite machine as shown below:

```
wesosm run infile <location of osdef.conf> -verbose
```

The utility will now proceed to connect to the Web server to query its file system. It will subsequently connect to the Policy Director server to create entries in the object space underneath **/WebSEAL-Lite/reverse/books.com.** If a server definition does not have a **query_command** associated with it, only the root branch will be created. After the object space has been created, you may place ACLs at the appropriate locations in it.

Over a period of time, the object space may become cluttered with obsolete entries that are no longer in use. To remove these obsolete entries from the object space, run the utility with the following parameters to remove the obsolete entries from the object space:

```
wesosm run infile <location of osdef.conf> -clean -verbose
```

The information covered thus far will provide a good foundation for building a more complex configuration. In the next section, a sample configuration will be designed for a simple Web site.

5 Deploying WebSEAL-Lite

5.1 Designing the Web Site

In this section, a complete WebSEAL-Lite configuration for **books.com** will be designed. This Web site will allow users to browse and purchase books. Many of the key features of WebSEAL-Lite will be illustrated in this example. The Web site design will be divided into the following components:

- **Content Distribution**
- **Single Sign-On**

Content Distribution

In this design, rather than storing the entire content for the Web site on one Web server, it will be distributed across several Web servers. It will be divided into the following sections:

- `/home`
- `/catalog`
- `/account`
- `/payment`

The **/home** directory will contain the greeting page, and links to other parts of the Web site. The **/catalog** directory will contain a repository of all the books sold at this Web site. These sections of the Web site will not be protected (*not require access control*).

The **/account** directory will contain HTML and Java code to manage the user accounts. The **/payment** directory will also contain HTML and Java code to receive the user payments for books to be purchased. Most

of these sections of the Web site will be protected. A directory underneath **/account** will be used to register new users. This directory will not be protected.

An alternative way of designing this Web site would be to assign a public sub-domain name to each section of the Web site. For example, instead of storing the catalog underneath **/catalog**, it would be stored on **catalog.books.com**. However, this method requires additional administration to allocate a sub-domain name for each section of the Web site.

Single Sign-On

In this design, an application running on the Web server hosting the **/account** directory will require an encrypted **LTPA Cookie** to identify the authenticated user. Another application running on the Web server hosting the **/payment** directory will require the HTTP header, ***App-User*** with the userid in it. It will also require basic authentication from WebSEAL-Lite as the trust basis for accepting the authenticated user. WebSEAL-Lite will be required to authenticate itself to this application with the userid, ***weblite***, and the password, ***bookworm***.

In this example, a relationship has been established with another vendor, **novels.com**, to forward authenticated users through WebSEAL-Lite to protected areas of **books.com**. The gateway at **novels.com** will be required to authenticate itself to WebSEAL-Lite using an authorization header with the userid, ***novelgateway***, and the password, ***bookworm***. The gateway will place the authenticated userid in the cookie, ***Novel-User***, as illustrated in Figure 2.

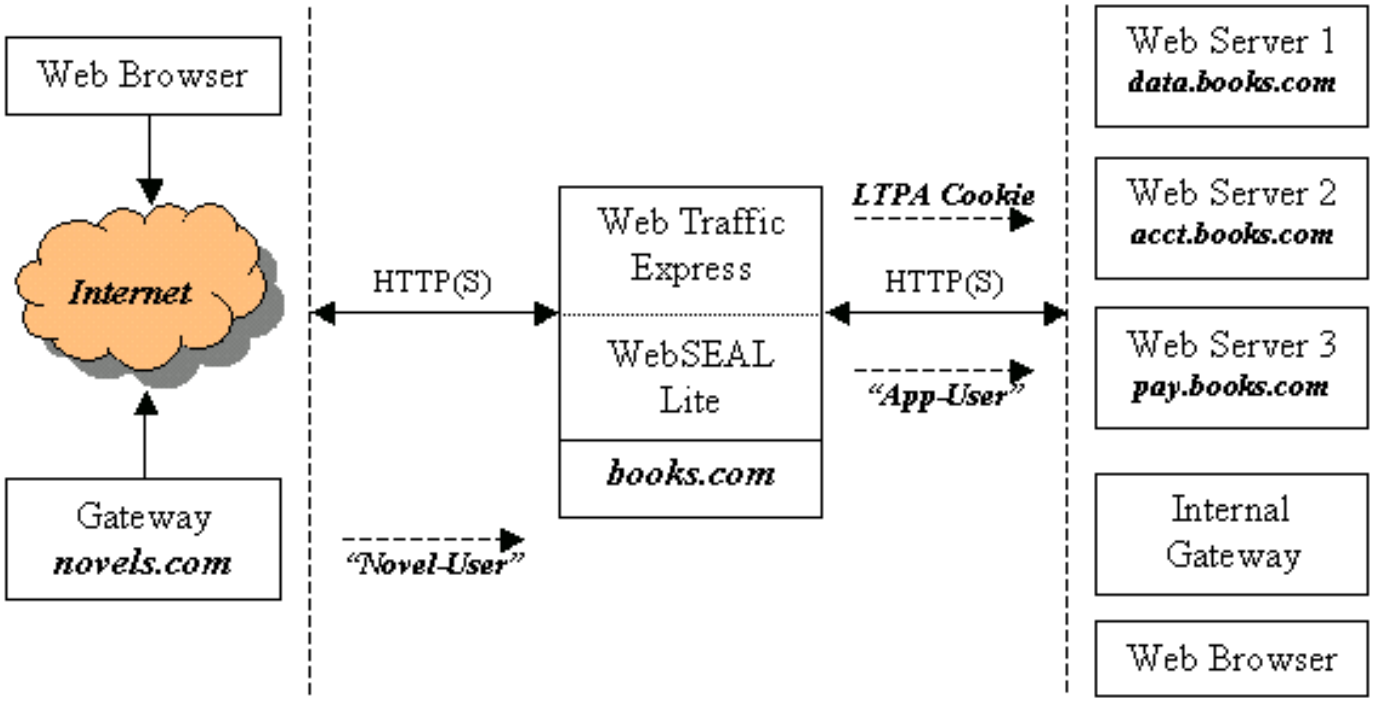

Figure 2 Web Site Design for **books.com**

5.2 Configuring the Web Site

In order to provide access control for **books.com**, WebSEAL-Lite will need to be configured. The configuration will begin by defining the global settings for WebSEAL-Lite, as shown below:

```
[Global]

# Administrator userid and password needed to run wesosm

update_admin_userid = sec_master
```

```
update_admin_password = secret5

# Error message indicating that SSL is required

require_ssl_errorfile = /opt/pdweb-lite/etc/require_ssl.htmls
```

The **[Global]** definition in the configuration file specifies settings that apply to every request handled by WebSEAL-Lite. In the above configuration, the administrator userid and password is provided so that the ***wesosm*** utility can create and update the object space. Also, if a request requires a secure connection and one is not provided, the specified error page will be returned to the user. However, if possible, the browser will automatically be redirected to the secure site.

The **[Local]** definition in the configuration file specifies settings that apply to objects on the file system managed by Web Traffic Express. There should only be one of these definitions in the configuration file. This server definition should already have been created by the configuration tool, and will be similar to the following:

```
[Local: /WebSEAL-Lite/bookproxy.com]

domains = bookproxy.com

query_command = http://bookproxy.com/cgi-bin/query_contents?dirlist=/
```

In this example, **bookproxy.com** is the primary domain name of the machine that Web Traffic Express is running on. The alias, **books.com**, is another domain name with its associated IP address, assigned to the same machine. WebSEAL-Lite differentiates between requests for objects belonging to Web Traffic Express and objects belonging to **books.com** by matching the requested domain name with a server definition in the configuration file. The **domains** attribute indicates which domains a server definition applies to.

The **[Remote]** definitions in the configuration file specify settings that apply to external Web servers. There is no limit on the number of server definitions you can have. For this example, the following definition would be appropriate for **books.com**:

```
[Remote: /WebSEAL-Lite/reverse/books.com]

domains = books.com www.books.com

# Form login and logout information

login_method = forms

form_login_file = http://books.com/home/login.html

form_login_errorfile = http://books.com/home/loginerr.html

form_logout_file = /account/logout.html

# Change password information

form_chgpasswd_file = http://books.com/home/chgpasswd.html

form_chgpasswd_submit_url = /account/chgpasswd

# Single sign-on tokens to look for
```

```
accept_sso = NovelSSO

# Server to map requests and initial page

# Browser will be redirected to greeting page

route = http://data.books.com/home /home/index.html
```

This configuration specifies the login method as forms, and lists the login forms. The login forms may be retrieved from a remote Web server (*begins with "http"*), or may be retrieved from the local file system. If the form has references to images in it, the URL links in the forms for these images should contain the full path of the images (*ie. "/home/gifs/banner.gif"*). The user will be logged out when the requested URLs path matches the path specified for the logout file. Also, as the configuration shows, a change password form will be sent to authenticated users when their passwords expire.

Single sign-on users will be accepted from the gateway at **novels.com**, using the **NovelSSO** single sign-on definition. This single sign-on definition must be defined to the configuration file. Finally, the initial request will be mapped to the **/home** section of the Web site, by redirecting the browser.

The **[Junction]** definitions in the configuration file specify settings that apply to virtual directories for this Web server. Each of these definitions represents a virtual directory that inherits its settings from its parent ([Remote]) definition as shown below:

```
[Junction: /WebSEAL-Lite/reverse/books.com:/home]

route = http://data.books.com/home

[Junction: /WebSEAL-Lite/reverse/books.com:/catalog]

route = http://data.books.com/catalog

[Junction: /WebSEAL-Lite/reverse/books.com:/account]

query_command = http://acct.books.com/cgi-bin/query_contents?dirlist=/

require_ssl = yes

submit_sso = LTPA-COOKIE

route = https://acct.books.com

[Junction: /WebSEAL-Lite/reverse/books.com:/payment]

query_command = http://pay.books.com/cgi-bin/query_contents?dirlist=/

require_ssl = yes

submit_sso = PayAppSSO PayAppAuth

route = https://pay.books.com
```

This configuration specifies the single sign-on tokens to submit to each Web server that requires one. It also tells WebSEAL-Lite to map the URL to the corresponding Web server hosting the requested content. If another routing module is to be used in place of the WebSEAL-Lite URL mapping, simply delete all references to the

route key word from the configuration file.

The **[SSO]** definitions in the configuration file define single sign-on tokens that may either be accepted or submitted as shown below:

```
[SSO: PayAppSSO]

type = header

name = App-User

format = <userid>

[SSO: PayAppAuth]

type = auth_header

format = weblite:bookworm

[SSO: NovelSSO]

type = cookie

name = Novel-User

format = <userid>

trust_basis = basic_auth

trust_list = novelgateway:bookworm
```

After these single sign-on definitions have been created, they may be supplied as parameters to the **accept_sso** and **submit_sso** key words. As illustrated in this configuration, all single sign-on requirements described in section **5.1**, have been satisfied. As you compare this configuration with the requirements listed in section **5.1**, you will find that WebSEAL-Lite is flexible and easy to configure.

5.3 Mastering WebSEAL-Lite

There are several other aspects of WebSEAL-Lite that may be configured. Accordingly, the sample configuration file documents each option in detail and provides a default and sample value for each option.

After developing a basic understanding of WebSEAL-Lite, you will be able to customize and configure it to work in your environment. Every option does need to be set to utilize the it effectively, since it will run well using the default values. Only the relevant options should be set.

When faced with the task of configuring WebSEAL-Lite, a good approach would be to clearly define the problem that needed to be solved, as illustrated in the previous section. Next, you would determine what features in WebSEAL-Lite would solve the problem. Finally, you would configure the corresponding configuration options with the appropriate values.