IBM Tivoli Access Manager for e-business: Junctions and Links

White Paper

Ori Pomerantz (orip@us.ibm.com)

June 2009

Copyright Notice

Note to U.S. Government Users—Documentation related to restricted rights—Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corporation.

Trademarks

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to valid intellectual property or other legally protectable right of Tivoli Systems or IBM, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user. Tivoli Systems or IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785, U.S.A.

Printed in Ireland.

Table of Contents

Introduction

White Paper

Conclusion

Introduction

About This Paper

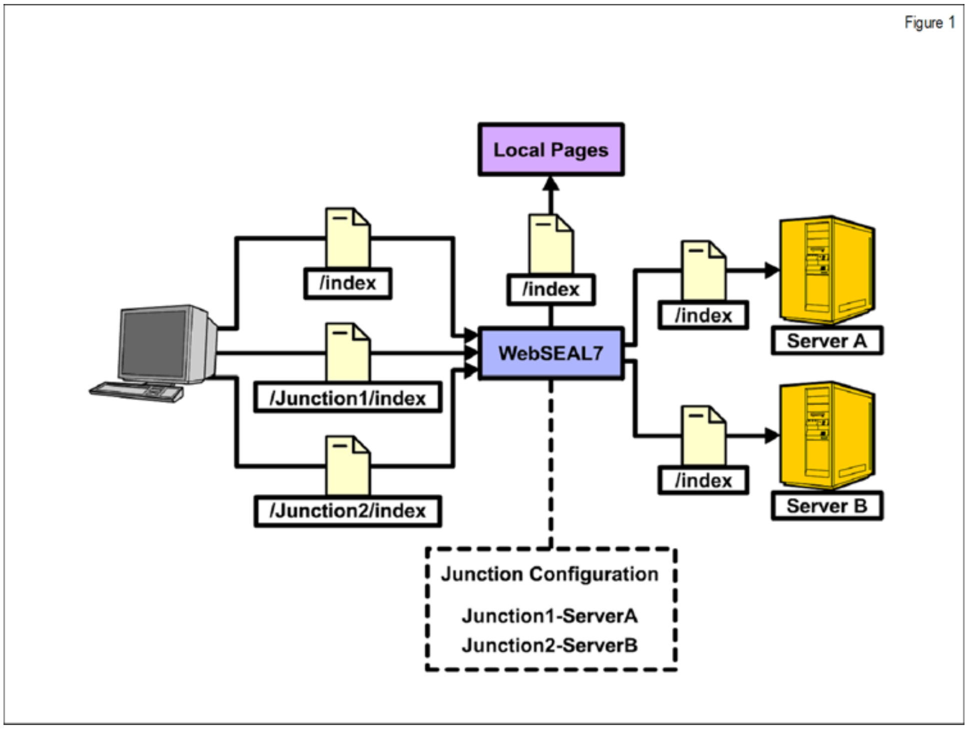
IBM® Tivoli® Access Manager for e-business uses junctions to identify Web servers. The client's browser uses one set of URL links and the back-end Web server uses another. This paper explains how WebSEAL processes URLs from the back-end server and the client.

Audience

This paper is intended to help implementors and system administrators who need to configure WebSEAL to protect back-end Web servers.

1 URLs, Links, and Junctions

WebSEAL is a proxy that is located between the Internet and the internal back-end servers. In the following figure, the browser requests a Web page using the URL **https://<webseal>/Junction1/index.html**. WebSEAL uses the junction name **Junction1** to identify the back-end server and sends a request for **http://ServerA/index.html**.



The problem is that Web pages contain links to other material (hypertext links, images, Adobe® Flash®, and so on). These links are all identified using URLs. The back-end server sends URLs that are valid for WebSEAL, but these URLs might not be valid for the browser because they do not contain the junction.

1.1 Link Types

There are three types of links:

- ***Relative*** links do not contain the name of the server or the name of the current directory. When the browser receives a relative link, the link appears to be located on the WebSEAL server. Relative links are correctly interpreted as links to other pages in the same directory on the same server.

  For example, assume that this line appears in **http://serverA/index.html**:

  ```
  <a href=”about.html”>About this site</a>
  ```

  The browser retrieved this page from **https://webseal/Junction1/index.html**. This URL is correctly interpreted as pointing to **https://webseal/Junction1/about.html**. This request would go back to WebSEAL and WebSEAL would know to request **http://serverA/about.html**.

- ***Server-relative*** links do not contain the name of the server, but they do contain the name of the directory.

  For example, assume that this line appears in **http://serverA/index.html**:

  ```
  <a href=”/contact.html”>Contact information</a>
  ```

  The browser retrieved this page from **https://webseal/Junction1/index.html**. This URL is interpreted as pointing to **/contact.html** on the same server. However, from the browser’s perspective the server is WebSEAL. If WebSEAL did not change the HTML, the browser would attempt to retrieve **https://webseal/contact.html** instead of the correct URL, which is **https://webseal/Junction1/contact.html**.

- ***Absolute*** links contain the name of the server and the directory.

  For example, assume that this line appears in **http://serverA/index.html**:

  ```
  <a href=”http://ServerA/copyright.html”>
  Copyright Information</a>
  ```

  If WebSEAL did not change the HTML, the browser would attempt to connect directly to **ServerA**, bypassing WebSEAL. A correctly configured firewall would only allow connections to **ServerA** from WebSEAL.

2 Modifying Outbound Links

As you have seen, server-relative and absolute links cannot work without changes. In some cases, WebSEAL can correct the HTML sent to the browser.

2.1 Links in HTML

WebSEAL automatically modifies links in HTML so that the browser will get URLs it can use:

- **Server-relative** links are modified to include the current junction name.
- **Absolute** links might or might not need modification. Only links to WebSEAL protected resources must be modified. WebSEAL changes the links to protected resources into server-relative links (by default) and adds the proper junction name. If the links are for external sites, WebSEAL does not change them.

The following example shows a fragment of an HTML page from a back-end server:

```
<A HREF=”about.html”>About this site</A></BR>
<A HREF=”/contact.html”>Contact information</A></BR>
<A HREF=”http://ServerA/copyright.html”>Copyright
information</A></BR>
<A HREF=”http://www.ibm.com”>IBM’s Web site</A></BR>
```

WebSEAL changes the links so that the browser receives this version:

```
<A HREF=”about.html”>About this site</A></BR>
<A HREF=”/Junction1/contact.html”>Contact information</A>
</BR>
<A HREF=”/Junction1/copyright.html”>Copyright information
</A></BR>
<A HREF=”http://www.ibm.com”>IBM’s Web site</A></BR>
```

2.2 Absolute Links and Canonical Host Names

WebSEAL requires the host name, *as encoded in the junction*, to be the same as the host name in the absolute link. When you create a junction, sometimes Tivoli Access Manager for e-business changes the server name to its canonical form. To change the server name to its canonical form, WebSEAL resolves the host name to an IP address and then resolves the IP address back to a name.

To identify this problem, view the junction using the following **pdadmin** command:

```
s t <webseal server> show /<junction>
```

The host name resolution occurs when the junction is created. If you cannot make the back-end server use the canonical name of the server in the absolute link, you can use the following solution:

1. Edit **/etc/hosts** to have a dummy host name that resolves to the name used by the back-end server. Add this line:

   ```
   172.16.0.1 <host name used by the back end> dummy
   ```

2. Create the new junction, using **-h dummy** as the host name.
3. Remove the line you added to **/etc/hosts**.

2.3 Links in Scripts

Typically, WebSEAL does not look in scripts to modify links. You can make WebSEAL modify absolute links within scripts using these steps:

1. Modify the WebSEAL instance configuration file. The **[script-filtering]** stanza must contain this line:

   ```
   script-filter = yes
   ```

2. Restart WebSEAL:

   ```
   pdweb restart
   ```

3. Create a junction with the junction cookie enabled (**-j** from the command line).

These steps turn on filtering, so that strings of the form **http[s]:**//<*host name*>/<*path*>, where the *host name* is a server in a junction, are modified to point to the correct junction.

Note: This method only works for absolute URLs, but it might not work in every case. Consider the following HTML coming from the back end:

```
<SCRIPT LANGUAGE=”JavaScript”>
<!--
document.write(“<A HREF=/bad.html>This will fail</A></BR>”);
var path = “ServerA/bad.html”;
document.write(“<A HREF=http://” + path + “>Link</A></BR>”);
document.write(“Go to http://ServerA/fun.html</BR>”);
// -->
</SCRIPT>
```

WebSEAL will modify this HTML and send the following HTML to the browser:

```
<SCRIPT LANGUAGE=”JavaScript”>
<!--
document.write(“<A HREF=/bad.html>This will fail</A></BR>”);
var path = “ServerA/bad.html”;
document.write(“<A HREF=http://” + path + “>Link</A></BR>”);
document.write(“Go to /Junction1/fun.html</BR>”);
// -->
</SCRIPT>
```

The first link will not be modified because it is server-relative. The second link, **http://ServerA/bad.html**, will also not be modified because WebSEAL will not be able to identify that it is a link. The string **http://ServerA/fun.html** *will* be modified even though it is not a link.

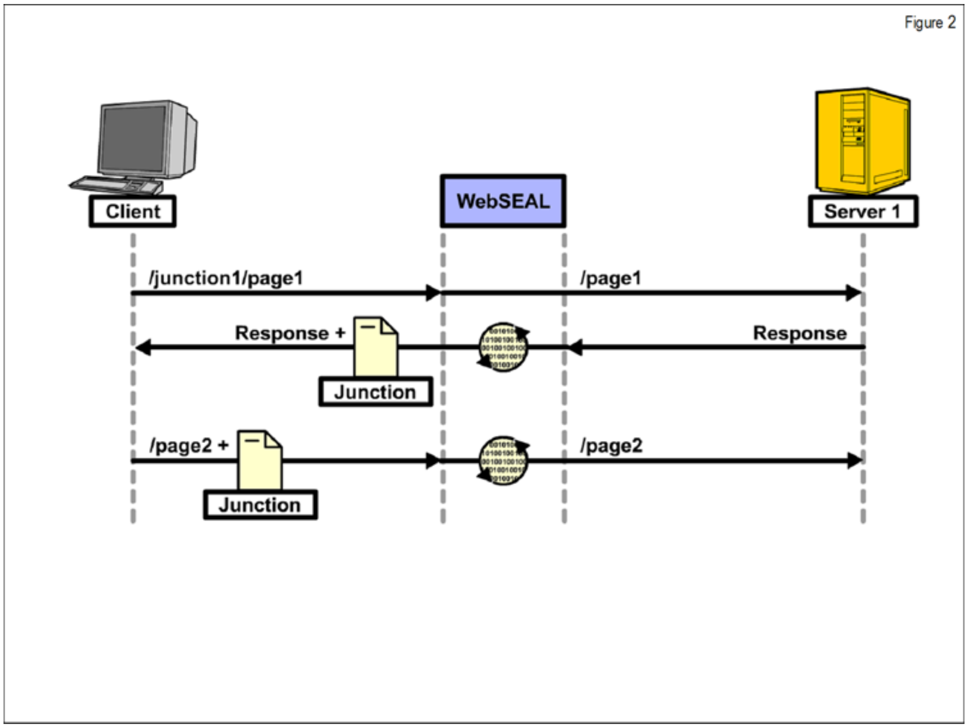
3 Inbound Server-relative Links

Server-relative URLs in scripts or applets that go through WebSEAL are not modified. Server-relative URLs do not encode the name of the server, so the browser sends the request back to WebSEAL. Although the page originally came from a back-end resource, identified by a junction, the browser operates as if it came from WebSEAL. Because the link is unchanged, the browser does not return the junction name.

WebSEAL can use several methods to identify the protected resource (or junction) that sent the script or applet to the browser.

3.1 Junction Cookies

If a junction is created with the **-j** option (enable junction cookie), WebSEAL adds JavaScript™ to every HTML page to include a cookie that contains the junction. When the browser requests another page from the same server, it sends back the cookie with the HTTP request.



The HTML source that WebSEAL sends to the browser starts with code such as:

```
<SCRIPT language=”JavaScript”>
<!--
document.cookie = “IV_JCT=%2FJunction2; path=/”;
//-->
</SCRIPT>
```

Using JavaScript, this code segment specifies that the cookie **IV_JCT** will be sent with any request for a page on this server that starts with a slash (/). The value of the cookie is **%2FJunction2**. The **2F** is the ASCII representation for /, so the value is an escaped **/Junction2**, the name of the junction.

This method fails in some cases. For example:

- If you keep a local copy of the page and click a link after the cookie expires, WebSEAL cannot direct the request.

- A different window or tab could overwrite the cookie if you perform the following steps:

  a. Open a page using a junction that has junction cookies enabled:

  ```
  https://<webseal>/Jct1/index.html
  ```

  The junction cookie is set to **Jct1**.

  b. In the same browser, open another window for a different junction on the same WebSEAL server, which also has junction cookies enabled:

  ```
  https://<webseal>/Jct2/page1.html
  ```

  The junction cookie is set to **Jct2**.

  c. When you return to the original window and click a link, for example to **/page2.html**, the cookie is set to **Jct2**. WebSEAL will attempt to retrieve **/page2.html** from the server for that junction, instead of **Jct1**.

- WebSEAL adds JavaScript to any page that the back-end server reports to be of type **text/html**. If the back-end server erroneously reports as HTML pages that are *not* HTML, WebSEAL adds JavaScript where it is not appropriate.

3.2 Referer in the HTTP Header

Some browsers report the ***referer***, the page that contains the link for the file the browser requests, in the HTTP header:

```
GET /page2.html HTTP/1.1
.
.
.
host: webseal
referer: https://webseal/Junction1/index.html
```

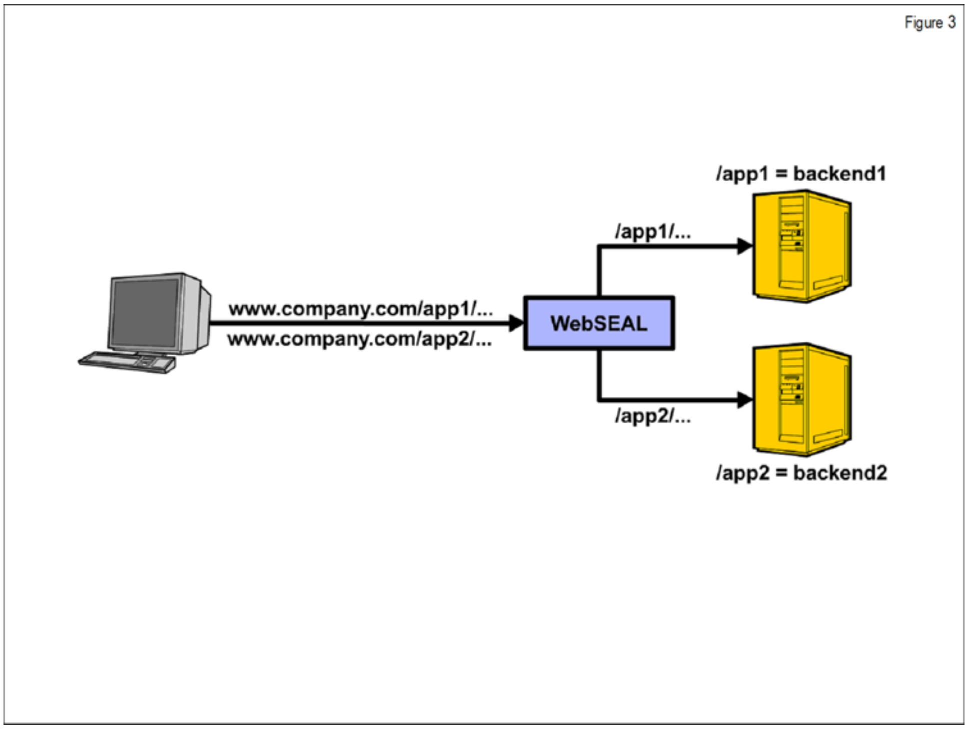
The first line of the request does not contain the junction. However, because the browser sent the referer header to WebSEAL, WebSEAL interprets the request as **https://webseal/Junction1/page2.html** and directs it to the correct back-end server.

Referer headers rely on the browser to send them out. Browsers do not always send referer headers.

3.3 Transparent Path Junctions

A ***transparent path junction*** has the same name as a directory on the back-end server. If different Web servers use different directories, you can use those directories as junctions.

In the following example, each back-end server has pages under a different directory. For example, **backend1** has an application under the **/app1** directory and **backend2** has a different application under the **/app2** directory.



Use the **-x** option to create the junctions:

```
s t <webseal server> create -t tcp -h backend1 -x /app1
s t <webseal server> create -t tcp -h backend2 -x /app2
```

Assume that WebSEAL receives a request for **https://<webseal>/app1/dir2/page3.html**. The junction is **/app1**, which is a transparent path junction. Therefore, WebSEAL directs a request to **backend1**, but does not remove the junction. The request goes to **http://backend1/app1/dir2/page3.html**.

Because WebSEAL does not change directory names in this scenario, server-relative links require no modification. However, this solution only works if the directory uniquely identifies the back-end server. If several back-end servers use the same directory name, transparent path junctions cannot be used because there is no way to know which back-end server is intended.

3.4 Junction Mapping Table

You can also manually specify regular expressions for URLs on the back-end server.

The ***junction mapping table*** is a text file that contains junctions and regular expressions. When WebSEAL looks for a junction, it tries to find which regular expression in the table matches. For example, if you have a Microsoft® Windows® server and a Linux® server, you might have the following junction mapping table:

```
/win *.asp
/win *.htm
/linux *.html
/linux *.php
```

The junction mapping table is located in **/opt/pdweb/www-default/lib/jmt.conf** by default. This file name is specified in the instance configuration file and can be modified as needed.

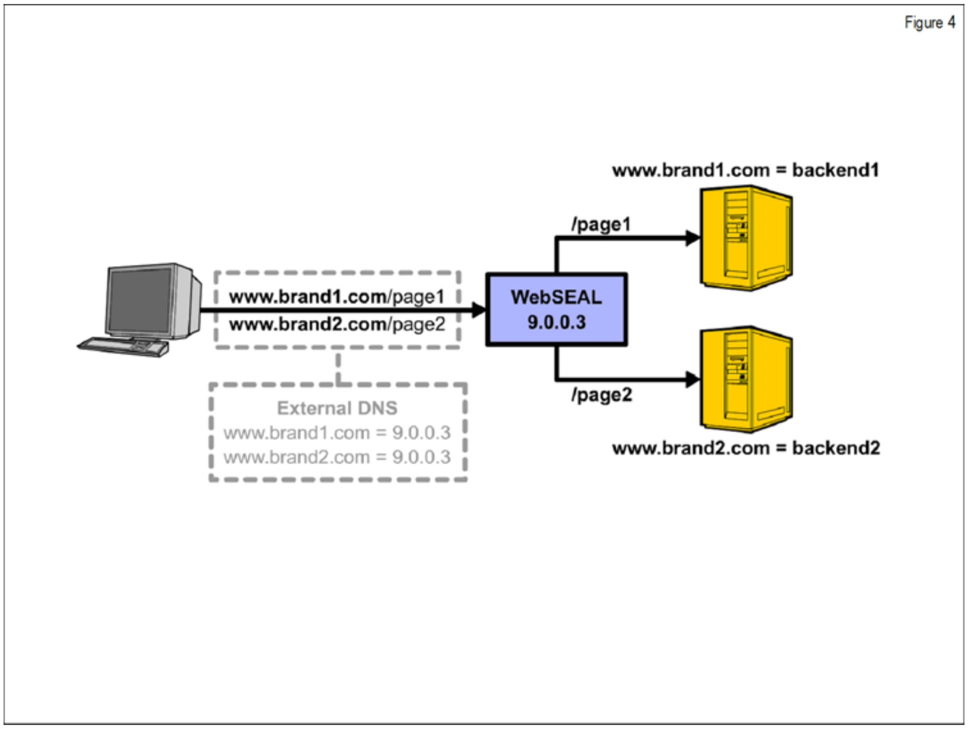
After you modify the junction mapping table, issue the following **pdadmin** command:

```
s t <webseal server> jmt load
```

4 Inbound Absolute Links and Virtual Host Junctions

Server-relative links at least have the advantage that the browser connects to the WebSEAL server to retrieve them. With absolute links, the browser attempts to resolve a different name to an IP address and to connect to that address.

The solution is to use ***virtual host junctions***. These are junctions that WebSEAL identifies using the **host:** HTTP header, instead of using a directory name. With virtual host junctions, multiple host names (for example, **www.brand1.com** and **www.brand2.com**) resolve to the IP address for WebSEAL.



When WebSEAL receives the request, the HTTP header contains a **host:** field that corresponds to the host part of the URL. For example, if the browser tried to retrieve **https://www.brand1.com/page1.html**, the HTTP request would look like the following example:

```
GET /page1.html HTTP/1.1
...
host: www.brand1.com
```

With this method, WebSEAL can receive absolute links and then deal with them correctly.

For directions on setting up virtual host junctions, go to the Tivoli IBM Education Assistant (IEA) site at **http://publib.boulder.ibm.com/infocenter/ieduasst/tivv1r0/index.jsp**. Click **Tivoli Access Manager for e-business > V6.1 >Junctions**. That page includes directions for TCP and SSL virtual host junctions.

Conclusion

Summary

Ideally, links in Web pages protected by WebSEAL should be relative links. However, WebSEAL is usually deployed in situations where the back-end server is not under the control of the same group. Often the back-end server is another company's product that cannot be modified by the organization.

Using the techniques in this paper, you can usually handle this situation with server-relative and absolute links.

Resources

- The HTTP protocol is defined in RFC 2616, available at the following URL:

  http://tools.ietf.org/html/rfc2616

- Cookies are explained in the draft specifications, available at the following URL:

  http://web.archive.org/web/20060424004149/wp.netscape.com/newsref/std/cookie_spec.html

- The Tivoli IEA site is at this URL:

  http://publib.boulder.ibm.com/infocenter/ieduasst/tivv1r0/index.jsp