An Oracle White Paper
March 2010

Using Oracle iPlanet Web Server as a Reverse Proxy for Improved Security

Executive Overview

Large-scale, highly distributed networked systems offer unrivaled efficiency and effectiveness for organizations by establishing extensive levels of organizational integration. However, accompanying the benefits of global integration are increased risks of invasion and compromise. Reverse proxies offer an effective method for protecting corporate Web infrastructures, such as Web sites, data, and applications. Oracle iPlanet Web Server can be configured as a reverse proxy, offering superior protection through hidden network topology, improved firewall effectiveness, and easy-to-enforce access control rules for all back-end Web servers.

Introduction

More and more attacks against corporate infrastructures occur via the HTTP layer. Standard security tools fail to analyze the HTTP protocol as an attack vehicle, but commercial reverse proxies offer an effective method for protecting Web sites and applications.

By definition, a proxy is a device that stands between two conversing entities. A forward proxy stands between a client and all other servers, such as in the case of outgoing internet requests from corporate employees. A reverse proxy does exactly the opposite. It stands between a server and all of its clients, such as in the case of incoming requests for a corporate Web site.

Oracle iPlanet Web Server can be configured as a reverse proxy to protect corporate Web infrastructures.

Employing Oracle iPlanet Web Server as a reverse proxy increases Web security through:

- **Single point of access, control, and logging.** Because all Web traffic flows through the reverse proxy, it is easy to enforce access control rules, such as those regarding access to IP addresses, for all back-end Web servers. Request logging is also centralized for improved monitoring. Note, however, that without failover, this creates a single point of failure, which should be avoided.

- **Hidden network topology**. Because reverse proxies map requests to content (for example, www.oracle.com/webserver to webserver.oracle.com), the corporate network topology is hidden from the outside world. This not only reduces the information vulnerable to attacks but also streamlines changes to the Web infrastructure.

- **Improved firewall effectiveness.** With the introduction of a reverse proxy, origin Web servers can be isolated behind the corporate firewall. In addition, the reverse proxy creates a new request to the origin Web server instead of passing the initial request.

Oracle iPlanet Web Server Reverse Proxy Plug-in

The reverse proxy plug-in is a Netscape Server Application Programming Interface (NSAPI) plug-in designed for use with the Oracle iPlanet Web Server 6.1 Service Pack 3 (SP3) and later service packs. This add-on enables Oracle iPlanet Web Server to act as a non-caching HTTP reverse proxy for specified uniform resource identifiers (URIs). A reverse proxy is a proxy that appears to clients to be a Web server (origin server) but actually forwards the requests it receives to one or more origin servers. Because a reverse proxy presents itself as an origin server, clients do not need to be configured to use a reverse proxy. If a given reverse proxy is configured to forward requests to multiple similarly configured origin servers, the reverse proxy can operate as an application-level software load balancer. A typical deployment has one or more reverse proxies deployed between browsers and origin servers.

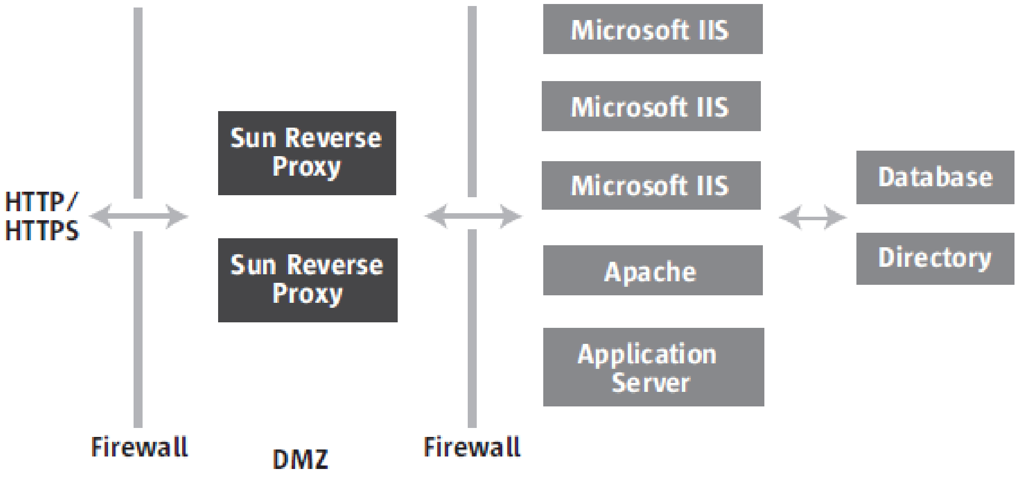

FIGURE 1: THIS SECURE WEB ARCHITECTURE INCLUDES A REVERSE PROXY.

The reverse proxy plug-in can be used in conjunction with existing Oracle iPlanet Web Server features such as on-the-fly gzip compression, output filters, advanced access control lists, and so on. The reverse proxy plug-in includes support for the following features:

- HTTP/1.0 and HTTP/1.1 compliance
- Credential pass-through
- Authentication to origin servers
- Data encryption
- Session stickiness
- Simple load balancing
- Granular error logging

HTTP/1.0 and HTTP/1.1 Compliance

The reverse proxy plug-in issues HTTP/1.1 requests to origin servers and accepts HTTP/1.0 responses to requests. It does not upgrade incoming HTTP/1.0 requests to HTTP/1.1 in ways that are incompatible with HTTP/1.0 (for example, it does not add a "Transfer-Encoding: chunked" header to a request).

Credential Pass-through

The reverse proxy plug-in passes through Basic-Auth and Digest-Auth credentials presented by the client. It encodes client certificates from the client and presents them in proprietary headers that can be used by an appropriately coded application on the origin server.

Authentication to Origin Servers

The reverse proxy plug-in can be configured to present its own credentials to an origin server. The reverse proxy plug-in can present Basic-Auth or utilize a specified certificate nickname.

Data Encryption

The reverse proxy plug-in is able to use Secure Sockets Layer (SSL) v2, SSLv3, and Transport Layer Security (TLS) technology when making requests to origin servers.

Session Stickiness

The reverse proxy plug-in can be configured to recognize "sticky cookies" and can configure the name of the sticky cookies.

Simple Load Balancing

The reverse proxy plug-in distributes load to several configured origin servers.

Granular Error Logging

The reverse proxy plug-in takes advantage of Oracle iPlanet Web Server's granular error logging capabilities (config, failure, warning, fine, finer, and finest). See the Oracle iPlanet Web Server documentation for more details.

Server Application Functions

The reverse proxy plug-in provides the following server application functions (SAFs):

- auth-passthrough
- check-passthrough
- service-passthrough

Auth-Passthrough

The auth-passthrough AuthTrans SAF inspects an incoming HTTP request for client information encoded by a service passthrough function running on an intermediate server. The client information includes

- The IP address from which the request originated, as encoded in the Proxy-ip header
- The SSL/TLS session ID of the originating connection, as encoded in the Proxy-ssl-id header
- The SSL/TLS cipher presented by the originating client, as encoded in the Proxy-cipher, Proxy-keysize, and Proxy-secret-keysize headers
- The SSL/TLS client certificate presented by the originating client, as encoded in the Proxy-issuer-dn, Proxy-user-dn, and Proxy-auth-cert headers

When auth-passthrough detects encoded client information, it instructs the server to treat the request as if it had arrived directly from the originating client instead of via an intermediate server running service-passthrough.

The auth-passthrough SAF is optional. When used, auth-passthrough is employed on the server instance that receives the request forwarded by service-passthrough.

Because auth-passthrough makes it possible to override information that may be used for authentication, such as the IP address of the original request, it is important that only trusted clients and servers be allowed to connect to a server running auth-passthrough. As a minimal precaution, only servers behind a corporate firewall should run auth-passthrough; no internet-accessible server should run it. Furthermore, if this information about the originating client is not required, do not use auth-passthrough.

The following obj.conf snippet demonstrates the use of auth-passthrough (note that these lines are not indented in a real obj.conf):

```
<Object name="default">

AuthTrans fn="auth-passthrough"

...

</Object>
```

Check-Passthrough

The check-passthrough ObjectType SAF checks to see if the requested resource, such as the HTML document or GIF image, is available on the local server. If the requested resource does not exist locally, check-passthrough will set the type to indicate that the request should be passed to another server for processing by service-passthrough.

The check-passthrough SAF accepts the following parameter:

- **type** (optional). The type to use for files that do not exist locally. If not specified, it defaults to magnus-internal/passthrough.

Service-Passthrough

The service-passthrough Service SAF, which forwards a request to another server for processing, accepts the following parameters:

- **servers.** A "space-delimited list of servers" that receive the forwarded requests. Individual server names can have an optional http:// or https:// prefix to indicate the protocol or can be suffixed with a colon and an integer to indicate the port.
- **sticky-cookie** (optional). The name of a cookie that causes requests from a given client to "stick" to a particular server. Once a request containing a cookie with this name is forwarded to a given server, service-passthrough attempts to forward subsequent requests from that client to the same server by sending a JROUTE header back to the client. If not specified, sticky-cookie will default to JSESSIONID.
- **user** (optional). The username that service-passthrough uses to authenticate to the remote server via Basic-Auth. Note that "user" also requires the use of "password."
- **password** (optional). The password that service-passthrough uses to authenticate to the remote server via Basic-Auth. Note that "password" also requires the use of "user."
- **client-cert-nickname** (optional). Nickname of the client certificate that service-passthrough uses to authenticate to the remote server.
- **validate-server-cert** (optional). Boolean that indicates whether service-passthrough should validate the certificate presented by the remote server. If not specified, validate-server-cert will default to false.
- **rewrite-host** (optional). Boolean that indicates whether service-passthrough should rewrite the Host header sent to remote servers, replacing the local server's hostname with the remote server's hostname. If not specified, rewrite-host will default to false.
- **rewrite-location** (optional). Boolean that indicates whether service-passthrough should rewrite the Location headers returned by a remote server, replacing the remote server's scheme and hostname with the local server's scheme and hostname. If not specified, rewrite-location will default to true.
- **ip-header** (optional). Name of the header that contains the client's IP address, or "" if the IP address should not be forwarded. If not specified, ip-header will default to Proxy-ip.
- **cipher-header** (optional). Name of the header that contains the symmetric cipher used to communicate with the client (when SSL/TLS is used), or "" if the symmetric cipher name should not be forwarded. If not specified, cipher-header will default to Proxy-cipher.
- **keysize-header** (optional). Name of the header that contains the symmetric key size used to communicate with the client (when SSL/TLS is used), or "" if the symmetric key size name should not be forwarded. If not specified, keysize-header will default to Proxy-keysize.
- **secret-keysize-header** (optional). Name of the header that contains the effective symmetric key size used to communicate with the client (when SSL/TLS is used), or "" if the effective symmetric key size name should not be forwarded. If not specified, secret-keysize-header will default to Proxy-secret-keysize.

- **ssl-id-header** (optional). Name of the header that contains the client's SSL/TLS session ID (when SSL/TLS is used), or "" if the SSL/TLS session ID should not be forwarded. If not specified, ssl-id-header will default to Proxy-ssl-id.
- **issuer-dn-header** (optional). Name of the header that contains the client certificate issuer DN (when SSL/TLS is used), or "" if the client certificate issuer DN should not be forwarded. If not specified, issuer-dn-header will default to Proxy-issuer-dn.
- **user-dn-header** (optional). Name of the header that contains the client certificate user DN (when SSL/TLS is used), or "" if the client certificate user DN should not be forwarded. If not specified, user-dn-header will default to Proxy-user-dn.
- **auth-cert-header** (optional). Name of the header that contains the DER-encoded client certificate in Base64 encoding (when SSL/TLS is used), or "" if the client certificate should not be forwarded. If not specified, auth-cert-header will default to Proxy-auth-cert.

When multiple remote servers are configured, service-passthrough chooses a single remote server from the list on a request-by-request basis. If a remote server is unavailable or returns an invalid response, service-passthrough will set the status code to 502 Bad Gateway and will return REQ_ABORTED, which will return an error to the client. This error is customizable in Oracle iPlanet Web Server by configuration of a customized response for the 502 error code.

When a user and the password are specified, service-passthrough uses these credentials to authenticate to the remote server, using HTTP basic authentication. When one or more of the servers in the servers parameter has an https://prefix, client-cert-nickname specifies the nickname that the client certificate service-passthrough uses to authenticate to the remote server.

Note that service-passthrough generally uses HTTP/1.1 and persistent connections for outbound requests, with the following exceptions:

- When forwarding a request with a Range header that arrived via HTTP/1.0, service-passthrough issues an HTTP/1.0 request. This happens because the experimental Range semantics expected by Netscape HTTP/1.0 clients differ from the Range semantics defined by the HTTP/1.1 specification.
- When forwarding a request with a request body, such as a POST request, service-passthrough does not reuse an existing persistent connection, because the remote server is free to close a persistent connection at any time and service-passthrough does not retry requests with a request body.

In addition, service-passthrough encodes information about the originating client in the headers named by the ip-header, cipher-header, keysize-header, secret-keysize-header, ssl-id-header, issuer-dn-header, user-dn-header, and auth-cert-header parameters (removing any client-supplied headers with the same name) before forwarding the request. Applications running on the remote server can examine these headers to extract information about the originating client.

Installing the Reverse Proxy Plug-in

The reverse proxy plug-in is available for use with Oracle iPlanet Web Server 6.1 SP3 or later service packs.

Package Contents

The contents of the platform-specific packages are

- Oracle Solaris, Linux, AIX:
  - README.txt
  - libpassthrough.so (the NSAPI shared object, or "plug-in")
- HP-UX:
  - README.txt
  - libpassthrough.sl (the NSAPI shared object, or "plug-in")
- Windows**:**
  - README.txt
  - passthrough.dll (the NSAPI shared object, or "plug-in")

Installing on Oracle Solaris, Linux, HP-UX, and AIX, Using tar Packaging

```
$ gzip-d sun-webserver61-passthrough-{sol|lin|hpux|aix}.tar.gz

       ;; Uncompress the tar archive,

       ;; where {sol|lin|hpux|aix} reflects

       ;; the operating system

       ;; environment the library will be used

$ tar xvf sun-webserver61-passthrough-{sol|lin|hpux|aix}.tar

       ;; Extract the tar archive
```

See the "Configuring the Reverse Proxy Plug-in" section.

Installing on Windows

```
$ unzip sun-webserver61-passthrough-win.zip

       ;; Uncompress the ZIP archive
```

See the "Configuring the Reverse Proxy Plug-in" section.

Installing on Oracle Solaris, Using SVR4 Packaging

```
$ su
```

```
       ;; root access is required to install
       ;; SVr4 packages
$ cd <path/to/package>
       ;; Change directory to where the package
       ;; is located
# pkgadd-d .
       ;; Install SUNWwbsvr-passthrough.pkg package
```

This places the shared object and README in /opt/SUNWwbsvr/plugins/passthrough.

See the “Configuring the Reverse Proxy Plug-in” section.

Installing on Linux, Using RPM Packaging

```
$ su
       ;; root access is required to
       ;; install RPM packages
$ cd <path/to/package>
       ;; Change directory to where
       ;; the package is located
# rpm-iUvh sun-webserver-passthrough.rpm
       ;; Install
       ;; sun-webserver-passthrough.rpm
       ;; package
```

This places the shared object and README in /opt/sun/webserver/plugins/passthrough.

See the “Configuring the Reverse Proxy Plug-in” section.

Configuring the Reverse Proxy Plug-in

The reverse proxy plug-in needs to be initialized in the Oracle iPlanet Web Server magnus.conf file and configured in the corresponding obj.conf file.

Magnus.Conf

</path/to/sharedobject> is the install path of the shared object, including the shared object itself.

The path elements are "/" regardless of the operating system.

```
Init fn="load-modules" shlib="</path/to/sharedobject>
```

Obj.Conf

Configuration of obj.conf varies, depending on the intended use. See the Oracle iPlanet Web Server documentation for the use and the syntax of obj.conf.

Example 1

This configuration will proxy the URI "/example" if it does not exist locally. A local copy of "/example" is preferred to a remote copy:

```
<Object name="default">

# Assign the URI "/example" (and any more specific URIs;

# /example/foo.html, /example/qwe.jsp, etc) the object name

# "server.example.com"

NameTrans fn="assign-name"

       from="/example(|/*)"

       name="server.example.com"

...

</Object>


# Execute these instructions for any resource with the assigned name

# "server.example.com"

<Object name="server.example.com">


# Check to see if a local copy of the requested resource exists. Only

# proxy the request if there is not a local copy.

ObjectType fn="check-passthrough"

type="magnus-internal/passthrough"


# Proxy the requested resource to the URL

# "http://server.example.com:8080" only if the "type" has been set to

# "magnus-internal-passthrough"

Service type="magnus-internal/passthrough"
```

```
       fn="service-passthrough"

       servers="http://server.example.com:8080"

</Object>
```

Example 2

This configuration proxies all requests for the URI "/app" without first checking for a local version. The reverse proxy plug-in provides its own credentials via Basic-Auth to the origin server.

```
<Object name="default">

# Assign the URI "/app" (and any more specific URIs;

# /app/foo.html, /app/qwe.jsp, etc) the object name

# "server.example.com"

NameTrans fn="assign-name"

       from="/app(|/*)"

       name="server.example.com"

...

</Object>


# Execute these instructions for any resource with the assigned name

# "server.example.com"

<Object name="server.example.com">


# Proxy the requested resource to the URL

# "http://server.example.com:8080"

Service fn="service-passthrough"


       servers="http://server.example.com:8080"

       user="blues"

       password="j4ke&elwOOd"


</Object>
```

Conclusion

Since their inception more than 40 years ago, networks not only have global reach but also have an impact on virtually every aspect of the day-to-day process of doing business. In fact, network systems are so fundamental to business, industry, and government that a secure platform is now crucial, whether for the protection of proprietary business data or functionality sitting in a demilitarized zone. Reverse proxies offer an effective method for protecting Web infrastructures, including corporate infrastructures. Oracle iPlanet Web Server can be configured as a reverse proxy, offering superior protection for organizations of any size, reach, and economic sector.

Appendix A: Tuning the Keep-Alive Subsystem in Oracle iPlanet Web Server 6.1

If you are using Oracle iPlanet Web Server 6.1 SP3 or higher, there is no need to adjust the keep-alive subsystem. Other than the timeout value, Oracle iPlanet Web Server 6.1 SP3 automatically selects the keep-alive threading model that is most efficient for the number of connections it is handling.

For versions prior to 6.1 SP3, the following information focuses on how to tune the keep-alive subsystem. For more-detailed information on tuning other parameters, see these documents:

- *Sun Java System Web Server 7.0 Update 7 Performance Tuning, Sizing, and Scaling Guide* at http://docs.sun.com/app/docs/doc/821-0786/abyat?l=en&a=view&q=Web+Server+7++Performance+Tuning
- *Sun ONE Web Server 6.1 Performance Tuning, Sizing, and Scaling Guide* at http://docs.sun.com/source/817-1836-10/perftune.html
- iPlanet Web Server 6.0 Performance Tuning, Sizing, and Scaling Guide at http://docs.sun.com/source/816-5690-10/perf6.htm

Keep-Alive Subsystem

Under certain conditions, the performance of a version of Oracle iPlanet Web Server can be slower than that of earlier versions (for example, 6.0 versus 4.1). This behavior is particularly evident when the Oracle iPlanet Web Server serves a page with many inline images to only a few clients. The root cause of this slow performance is the scalable keep-alive subsystem. When Oracle iPlanet Web Server 6 is configured in the standard, out-of-the-box installation, its performance can be suboptimal in low-load conditions that rely heavily on keep-alive connections.

Performance Parameters

This section explains the functions and settings of the keep-alive performance parameters, which reside in the magnus.conf file.

TABLE 1. FUNCTIONS AND ATTRIBUTES OF THE KEEP-ALIVE PERFORMANCE PARAMETERS

PARAMETER	FUNCTION	ATTRIBUTES
MaxKeepAliveConnections	This parameter controls the maximum number of keep-alive connections Oracle iPlanet Web Server can maintain at any time.	Default, 256; range, 0–32,768
KeepAliveTimeout	This parameter controls the maximum time, in seconds, that Oracle iPlanet Web Server holds open an HTTP keep-alive connection or a persistent connection between the client and the server.	Default, 30 seconds; maximum, 300 seconds (5 minutes)
KeepAliveThreads	This parameter controls the number of threads in the keep-alive subsystem. Oracle	Default, 1

	recommends that you adjust this number to be a small multiple of the number of processors on your system. For example, assign two or four keep-alive threads to a two-CPU system.	
KeepAliveQueryMeanTime (in Oracle iPlanet Web Server 6.0, Service Pack 2, or above only)	This parameter specifies the desired keep-alive latency, in milliseconds. On lower-load systems, you can lower this number to enhance performance. Doing so can increase CPU usage, however.	Default, 100
KeepAliveQueryMaxSleepTime (in Oracle iPlanet Web Server 6.0, Service Pack 5, or above only)	This parameter sets an upper limit on the time slept (in milliseconds) after polling of keep-alive connections for further requests. On lightly loaded systems that primarily service keep-alive connections, you can lower this number to enhance performance. Doing so can increase CPU usage, however.	Default, 100 milliseconds; range, 0–5,000 milliseconds

KeepAliveQueryMeanTime and KeepAliveQueryMaxSleepTime control latency. Changing their values affects Oracle iPlanet Web Server as follows:

- By lowering their settings, you lower the latency of lightly loaded systems. An example is reduced page load times.
- By raising their settings, you raise the aggregate throughput of heavily loaded systems. An example is an increased number of requests the server can handle. However, in the case of too much latency and too few clients, the server remains unnecessarily idle and causes aggregate throughput to deteriorate.

Keep in mind the two general rules for tuning keep-alive subsystems for a particular load:

- In the event of idle CPU time, decrease the setting for KeepAliveQueryMeanTime and KeepAliveQueryMaxSleepTime.
- In the event of no idle CPU time, increase the setting for KeepAliveQueryMeanTime and KeepAliveQueryMaxSleepTime.

Oracle is committed to developing practices and products that help protect the environment

Using Oracle iPlanet Web Server as a
Reverse Proxy for Improved Security
March 2010

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com