PROJECT LOCKDOWN – OHS WEB SERVER EDITION

Brian Mulreany, Unisys
Kevin Sheehan, Agilex Technologies

EXECUTIVE SUMMARY

This paper provides practical solutions for securing your Oracle HTTP Server (OHS). It follows the approach used with database Project Lockdown by Arup Nanda, to provide a phased approach to securing the web server. The introduction explains the differences in web server setup based on usage, from reverse proxy server to J2EE Application server front-end. The presentation then walks through each of the phases providing specific details in the form of scripts and tools that can be used to execute the task. Phase 1 covers the basic setup and hardening that should be done as part of the install. Phase 2 focuses on advanced hardening and verifying the setup using Grid Control. Phase 3 identifies what to look for when monitoring the web server, and how to set appropriate policy monitoring. Phase 4 looks at the 11g Application Server and new features and capabilities that should be considered with an 11g upgrade. The References section at the end of this document contains Metalink notes, web sites, and tools to assist in locking down the web server.

Contents:

SECURITY ARCHITECTURE OVERVIEW

While securing your web server is critical, doing so without consideration of the security posture of the entire environment is foolhardy. This section introduces the concept of *defense in depth* to ensure all potential attack vectors are considered.

CONFIDENTIALITY, INTEGRITY & AVAILABILITY (CIA)

The goal of the security architecture is simply to protect the confidentiality, integrity and availability of the information system itself, the data within it and the security controls that protect it.

- Confidentiality – Information should only be disclosed to authorized systems or personnel. Common controls used to enforce confidentiality include encryption and *role based access controls* (RBAC). Examples of common breeches of confidentiality include: sharing passwords, shoulder-surfing, sniffing unprotected wireless networks and stolen laptops and/or removable media.
- Integrity – Protecting information from unauthorized changes. Common controls used to enforce integrity include enforcement of a *least privilege* access control model, regular job rotations and technical controls such as referential integrity, anti-virus software and *Virtual Private Database* (VPD). Examples of common breeches include: computer viruses and SQL Injection attacks.
- Availability – Ensuring the system, as well as the access, audit and other security controls that protect the system, are continuously available as per business requirements. Common controls used to enforce availability include redundant hardware and software, *Oracle Real Application Clusters* (RAC) and disaster recovery sites. Examples of common breeches of availability include *Denial of Service* (DoS) attacks and single points of hardware or software failure.

Since the October 2006 *Critical Patch Update* (CPU), Oracle has used the *Common Vulnerability Scoring System* (CVSS) to evaluate the impact of Oracle security vulnerabilities on CIA. It is strongly recommended that information technology organizations that use Oracle products analyze the quarterly Oracle CPUs, their associated CVSS scores and their own security controls in order to determine an appropriate patching strategy. For more information on CVSS see http://www.first.org/cvss/cvss-guide.html and the CVSS calculator at http://nvd.nist.gov/cvss.cfm?calculator&version=2 .

DEFENSE IN DEPTH

Defense in depth has its origins in military strategy. Conventional military theory amassed troop concentrations along the front line of the battlefield but if the enemy broke through that line there were no other defenses available. A defense in depth strategy spreads forces out into successive lines of protection well before the front line of battle. The strategy also employs multiple technologies in anticipation of differing military attack strategies, for instance, interspersing anti-tank trenches to stymie armored attacks or lines of barbed wire to slow infantry attacks. The strategy allows defensive troops to fall back in a controlled manner, thus slowing down the enemy and giving military commander's time to determine the location of the attack and respond accordingly with additional military support.

Defensive strategies have matured within information security in a similar fashion. Originally, information security concentrated on protecting the perimeter, or edge, of your environment with firewalls and possibly *Network Intrusion Detection Systems* (NIDS) which was tantamount to putting all your defenses on the front line. Such a strategy is often called "crunchy on the outside, soft on the inside." It has the same pitfalls as the similar military strategy in that once the perimeter is breeched; there are no other defenses available to protect your systems. It also provides no protection from security breeches INSIDE the firewall, and current published estimates report that inside attacks account for 70-90% of all incidents.

Within information security, defense in depth represents a layered approach to security by addressing the security posture of not only the technology stack but the operational and management controls within the system as well. A comprehensive list of controls can be found in the *National Institute of Standards and Technology* (NIST) "Recommended Security Controls for Federal Information Systems", aka NIST 800-53: http://csrc.nist.gov/publications/nistpubs/800-53-Rev3/sp800-53-rev3-final-errata.pdf.

Since the focus of this paper is on the technology stack, we recommend at minimum the following technical controls within your system(s):

- Use firewalls to carve out, at minimum, three zones of protection:

- *Demilitarized Zone* or DMZ which will typically contain your proxy servers
- Intranet or trusted zone that will contain your application servers and potentially your end-users
- Database or restricted zone that will contain your database servers
- Use at least two different types of firewall technologies so that a breech of a single firewall does not compromise all the security zones
- Use separate network infrastructure (routers, switches, etc.) for internet-facing components in the DMZ from that used within your intranet
- Use a switched network to restrict traffic to point-to-point communication
- Deploy NIDS
- Deploy a *Host Intrusion Detection System* (HIDS) on all servers
- Centrally manage NIDS and HIDS, typically from a *Security Operations Center* or SOC which is responsible for security incident management
- Harden all components (network, servers, web servers, application servers and database servers) based on secure benchmarks such as those provided by the *Center for Internet Security* (CIS) at http://www.cisecurity.org
- Insure all inbound http traffic flows through a reverse proxy server and all outbound http traffic flows through a forward proxy server
- Use redundant servers at all tiers including RAC at the database tier
- Encrypt all end user web traffic, preferably with a hardware-based SSL accelerator, typically deployed at the load balancers in front of the web or reverse proxy servers.
- Use only *Federal Information Processing Standard* FIPS 140-2 certified cryptographic modules for encryption. A list of vendors using such modules can be found at http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401vend.htm.
- Do not allow end-users to directly access the database. Such access should only be available through an application.
- Harden your database listener via by turning on valid node checking and providing a list of invited nodes that is restricted to your application servers

INTRODUCTION

ROLE OF THE WEB SERVER

The web server is a critical component in the security architecture for a number of reasons. First, it is the gateway to all of your web-based applications and thereby the de facto attack vector of choice for those who intend to do harm. Second, the web server, by default, is configured to serve content and not protect content. Unless you take action to properly lock it down, OHS is insecure by default. The general strategy for protecting content is first lock it all down and then open only what should be opened. And finally, OHS contains a number of capabilities such as reverse proxy and mod_security that can significantly enhance the security posture of your system(s).

HOW SHOULD THE WEB SERVER BE CONFIGURED FOR EACH USE CASE?

If you are running a large web site with multiple Oracle products then it is likely that you have multiple versions of OHS in operation supporting different products. If you are running the E-Business Suite (EBS), then you have a version of OHS using Apache 1.3. If you have 10gAS installed, then you have a different version of Apache 1.3. If you are running Apex applications or you have implemented a reverse proxy, then you probably have implemented the OHS version based on Apache 2.0, since this provides unique benefits over Apache 1.3. You may also be using Universal Content Management with an OHS based on Apache 2.0 or running OBIEE with a 10.1.3 admin install with a slightly higher version of Apache 1.3.

There are major differences between Apache 1.3 and Apache 2.0, which means you cannot use a standard configuration file for all the products. The E-Business suite setup is typically managed with autoconf. While it is possible to use a custom httpd.conf file with EBS, this can create other complications. You probably want to plan on running three primary configurations, one for Apache 1.3, one for Apache 2.0, and a third for EBS. Within these configurations you can still use a similar setup for minimizing information leaks, such as standard server directive settings or custom error pages. There will still be differences based on the functions that you are supporting. We would recommend that you tailor the base configuration for the specific functions that you support. If you are using OHS as an Apex front-end, then there is no need for a lot of load modules and these should be disabled. We would recommend that you follow the axiom form follows function and adapt the techniques described in this paper to your requirements and the products you support.

DAY

PHASE 1: SETTING UP THE WEB TIER

FIRST THINGS FIRST – EXACTLY WHAT SHOULD I INSTALL ON MY WEB TIER?

It seems a trivial question but it turns out to be a little more difficult than you might think and yet the answer is critical to your success. Let's begin with our first OHS lesson learned by reviewing the evolution of the web tier in our own environment.

HOW OUR WEB TIER EVOLVED

Our environment is entirely Oracle-based, including the web, application and database tiers. We support custom J2EE applications and Oracle Portal, so our middle tier has a J2EE/Portal installation of the Oracle Application Server.

CHIMPS (OR IS THAT CHUMPS)

Our initial designed called for Oracle Web Cache at the web tier. So what did we install? We did a full J2EE/Portal install and then configured and started only Web Cache. Seems silly now but we thought it would be easier to simply clone the middle tier. This made everything the same on both the web and application tiers and the thinking was that this would be easier to maintain.

In reality we just created more headaches for ourselves. With this heavy a footprint, patching was a constant issue. In addition, we had violated a fundamental security policy by installing software that was not needed. To make matters worse, this software was installed in our DMZ.

NEANDERTHALS

Our next evolutionary step was to replace the full blown J2EE/Portal install with Standalone Web Cache. This improved our security posture by significantly reducing our footprint. Patching was much quicker since only a single component was installed. However, we started having misgivings about Web Cache itself.

First of all, Web Cache is a single-thread process[1]. Oracle recommends a 2-CPU server, one for the *Operating System* (OS) and one for Web Cache. Additional CPUs will have no impact; that is, you cannot scale up. If you need additional throughput you must buy more servers.

Second, Web Cache cannot be configured as a reverse proxy nor as a software firewall such as mod_security and it cannot be used to rewrite URLs.

HOMO SAPIENS

We have now replaced Standalone Web Cache with Standalone OHS. In the initial rollout, OHS was configured as a reverse proxy server. This allowed us to block any URLs that were not valid on the platform and stopped a lot of internet probing of the system by bad guys trying to *fingerprint* the environment.

In a second phase we deployed mod_security to block common signatures for *Cross-Site Scripting* and *SQL Injection*.

Note also that we deployed the Apache 2.0 version of OHS that allowed us to use the threaded implementation called *MPM Worker*. This gave us much greater throughput without buying any new hardware.

This configuration gives us better security and better performance, a rare win-win.

[1] Note that this discussion is based on the 10g version of Web Cache. 11g Web Cache has fixed some of these issues.

Let’s grab an Oracle DVD and get started with installing OHS

There are 10+ versions of OHS

Oracle distributes over 10 different variations of OHS. This includes distributions with the Application Server, E-Business Suite, and the Database. The different versions are based on two different versions of Apache, 1.3 and 2.0. Oracle has released a new version of the Application server based on Apache 2.2.13. The 2.2 version of Apache modified the Apache architecture. Load modules used with earlier versions of Apache are not compatible. This means a third major version will be part of the picture. Within each of these main distributions there are different point releases that correspond to the products and product releases that include OHS. This means 10gAS has slightly different versions depending on the 10gAS release. The same is true for EBS and the Database. Even Oracle has a hard time keeping track of all these versions. One of the OHS Metalink notes has this statement about an OHS version: *“It is externally labeled as "10.1.3.3", but the component version is actually "10.1.3.1", and is a special build, different than the Oracle Application Server counterpart. “* This can complicate maintenance, and particularly patching of OHS to prevent security vulnerabilities. You may find that the architecture goal of using the best Oracle technology available competes with the security goal to minimize the number of versions of products in production. When planning an OHS deployment you need to balance these goals to produce the best solution. If you have an unlimited maintenance budget, or if using every OHS version that has ever been released is in your bucket list, then just grab the first download and get started. If not, you need to carefully consider which versions you plan to use.

All OHS versions are not created Equal

Did you see this statement from Metalink note # 400010.1 - Steps to Maintain Oracle Database 10.2 Companion CD Home:

> ***“Something to think about...***
>
> *The Oracle HTTP Server delivered with the Oracle Database 10.2 Companion CD is provided to initially get HTMLDB installed and running. However, its an older version with limited functionality and support. Both the Oracle HTTP Server and HTMLDB from this CD would need to be upgraded at this time. The Companion CD also installs a mix of 10.2 and 10.1 products which is more difficult to maintain.”*

It’s certainly convenient if you need OHS to install the version that comes on the database companion CD. However, you should not do it. The main versions of OHS come packaged with the Application Server. Oracle does not recommend that you use the database-shipped version, so don’t! If you choose to use a standalone version of OHS take the extra step of downloading the version that comes with 10gAS. If you decide to go ahead and use the database version of OHS, please remember that OHS patches are not included in database patchsets. When you apply a patchset to your database home, you must separately apply any OHS patches.

Basic Oracle HTTP Server (OHS) Hardening

So you've done your homework and have determined the appropriate version of OHS to run in your environment. This section of the paper covers the minimum you must do to secure that OHS implementation; however, before you crank up the *Oracle Universal Installer* (OUI), make sure you can answer the following question.

How Many OS Accounts are Required to Run OHS?

Consider the *Two-Man Rule* or the *Four-Eye Principal* that requires two people to initiate a critical function such as launching a nuclear missile or approving a financial transaction over a certain dollar threshold. The same concept applies to operating system accounts. Wherever possible, limit the capabilities of an account be spreading responsibility across multiple accounts. In this way, compromise of a single account does not put your entire environment in jeopardy.

For OHS, then, this means we should use three accounts on the web server:

- One to own the software (ORACLE_HOME)
  ```
  -rwx------ 1 oracle oinstall 428250 2006-09-18 16:16 ../bin/httpd
  ```
- One to run the software (httpd)
  ```
  httpd.conf:
  User webohs
  Group webohs

  ps –ef | grep httpd
  root 25846 5810 27 23:19 ? 00:00:01 httpd -d
  webohs 25847 25846 0 23:19 ? 00:00:00 httpd -d
  ```
- One to own the content (htdocs)
- drwxr-xr-x 2 webdoc webdoc 4096 2009-03-09 23:20 webcontent
- Now that you've created those three OS accounts, go head and crank up OUI.

Secure httpd.conf Configuration

The Apache web server is a versatile product with lots of options to configure and supports a wide variety of web applications. It can act as a proxy server, directly run applications such as Perl and PHP, front-end a Java application server, or just serve up content.

This is like the all inclusive resorts such as Club Med. All the activities, food, and drink are available in one place. However, even these resorts modify their model to appeal to specific clientele. The resort locations and activities are designed to fit the groups they cater to. You can go to a resort that is setup for families, for couples, or for singles. While the overall experience is consistent with the resort's philosophy, the activities available at individual clubs can vary widely.

When you are configuring the web server to support applications are you setting the options to cater to your clientele? This can improve the overall security and performance of your site. The Apache server includes a large number of modules in the core distribution, and vendors that distribute Apache add even more. If you are running a large site, then you probably have many instances of Apache running supporting different needs. This would include different environments, prod and test, different access points, Internet and Intranet, and different functions like proxy and application front-end.

Managing different configurations for each use case would be time consuming and error prone. Apache includes a core feature called IfDefine to mark conditional directives. This can be used to control which directives apply based on a runtime parameter. Using this option will allow you to maintain a single configuration supporting multiple uses while improving the security of the site. Here are some examples of using this capability.

Set a command line option, RUNPROXY, to control enabling proxy LoadModules if this instance is used as a proxy. This technique is typically used for loading the SSL module.

```
<IfDefine RUNPROXY>
  LoadModule proxy_module modules/mod_proxy.so
```

```
  LoadModule proxy_connect_module modules/mod_proxy_connect.so
  LoadModule proxy_http_module modules/mod_proxy_http.so
  LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
</IfDefine>
```

Set a command line option, TESTENV, to allow display of Apache server status if this is a test environment:

```
<IfDefine TESTENV>
  <Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from .your_domain.com
  </Location>
</IfDefine
```

Set a command line option, INTRANET, to enable a specific virtual host on the Intranet:

```
<IfDefine INTRANET>
  <VirtualHost intranet.your_domain.com:80>
    Port 80
    DocumentRoot /intranet/docs
    TransferLog /intanet/logs/access.log
    ErrorLog /intranet/logs/error.log
  </VirtualHost>
</IfDefine>
```

The command line options you use can be tailored to fit your environment and requirements. You can maintain a master configuration file and still support a custom setup that is appropriate for the clientele you serve. We will utilize this technique in the next few sections to harden the OHS setup.

Configuring OHS using IfDefine

The OHS configuration is controlled by OPMN. In order to setup conditional IfDefine conditions you need to modify opmn.conf. The following change will start OHS with a command line variable to run a proxy:

```
  <process-type id="OHS" module-id="OHS2">
    <module-data>
      <category id="start-parameters">
        <data id="start-mode" value="ssl-enabled"/>
        <data id="command-line" value="-D RUNPROXY"/>
      </category>
    </module-data>
    <process-set id="OHS" numprocs="1"/>
  </process-type>
```

You need to stop and start OPMN to have the changes take effect. Once this is done if you check the process you will see the parameter with the ps command you will see that it is set: `httpd.worker -DSSL -DRUNPROXY`

Stop Leaking Configuration Information

To stop configuration information from leaking we are going to make several configuration changes. First, we will turn off Server Headers and Tokens to eliminate obvious places where the information is being shown. We will eliminate methods that are not required and that can reveal the type of web server. We will replace error pages with custom versions that provide only the basic response information. Finally, we will insert some fake headers that will modify the order of the headers that are sent and that may confuse a fingerprinter into thinking we are running some other version of software. The following table highlights the changes:

Table 1 - Configuration changes for server information

Original Configuration	Revised Configuration
`ServerAdmin you@example.com`	`###ServerAdmin you@example.com`
`ServerName bjm-desktop`	`ServerName ohs.collaborate10.org`
`ServerTokens Minimal`	`ServerTokens Prod` Note that *ServerTokens Prod* get you a lower CIS score but Oracle provides the option of *ServerTokens None*. You should use the Oracle-provided *None* for the highest security.

No Limits on GET and POST or OPTIONS	`<LimitExcept GET POST>`
	`deny from all`
	`</LimitExcept>`
	`Options None`

Using default error pages	`ErrorDocument 403 /error_contactus.htm`
	`ErrorDocument 500 "There was an error processing your request, please retry."`

No fake headers	`Header onsuccess set X-Powered-By "ASP.NET"` Unless you are actually running ASP.NET on your application tier in which case pick something else!

Limiting access to Directories and File Types

Limiting access to the Root Directory

Here is an example of the default configuration:

```
# Default controls on root directory
<Directory />
    Options FollowSymLinks MultiViews
    AllowOverride None
</Directory>
```

Note that the root directory is completely unprotected by default and OHS will happily server up whatever content is there. You need to lock this down. Here's what a locked down configuration looks like:

```
#
# Each directory to which Apache has access can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of features.
#
```

```
<Directory />
#
# The Options directive has been removed since specifying
# anything other than Options All causes 403 errors for
# all requests. Options All is the default.
#
     AllowOverride None
     Order deny,allow
     Deny from all
</Directory>
#
# Because of the issue above with Options in the Directory directive,
# we use a DirectoryMatch directive to accomplish the same thing.
#
<DirectoryMatch ^/>
     Options None
</DirectoryMatch>
#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below.
```

Note that we completely lock down root with *Deny from all* and later in httpd.conf we open up only what we need. Note also the odd issue with *Options*. For reasons unknown to us adding *Options None* to the root *Directory* directive (which is what we want to do) causes 403 errors. We got around this problem by instead specifying it with the *DirectoryMatch* directive. We have no explanation for this - just a heads up. Please contact the authors if you can explain this behavior.

Limiting Access to the Document Root

Here is the default configuration for htdocs:

```
DocumentRoot "/u01/app/oracle/product/10.1.3.2/companionCDHome_1/ohs/htdocs"
<Directory "/u01/app/oracle/product/10.1.3.2/companionCDHome_1/ohs/htdocs">
    Options FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

You should lock your content directory down as follows:

```
#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory /ORACLE_HOME/ohs/htdocs>
#
# Allow only Get and POST methods within DocumentRoot
#
      <LimitExcept GET POST>
           deny from all
      </LimitExcept>
#
# The Options directive is both complicated and important.  Please see
# http://httpd.apache.org/docs-2.0/mod/core.html#options for more information.
#
     Options None
#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
```

```
    AllowOverride None
#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all
</Directory>
```

Note that we restrict all methods except *GET* and *POST*. In addition we set *Options* to *None*. Finally, we are allowing everyone access to content on this server. If it makes sense for your environment you can lock down access to particular subnets or IP addresses with the *Allow* directive. Remember that we have locked down the root directory so you must open DocumentRoot or all access attempts will result in a 403 error.

Limiting access to CGI & perl Scripts

You should lock down access to CGI or perl scripts to only appropriate personnel. We do not even use CGI or perl so we don't load their corresponding load modules. Still, remembering Defense in Depth, we lock down access further since all it takes is one person to uncomment the cgi_module and then any such scripts are executable by anyone unless restricted as we do here. To avoid that single point of security failure we lock such scripts down as follows:

```
<LocationMatch /(cgi-bin|fcgi-bin|cgi|fastcgi/testperl\.pl|fcgi\.jar|libfcgi\.a)/>
    Order deny,allow
    deny from all
    allow from localhost
</LocationMatch>
```

Here we let the scripts execute only from localhost but you could eliminate them entirely by just removing the allow directive or saying allow from none.

Limiting access to Backup Files

Remembering the credo of "first do no harm", production support personnel should always backup configurations before changing them. Oracle will also make backup copies when applying patches. If these patches were security patches, that means the unpatched versions are sitting on your server and unscrupulous people will try to find and use them to exploit known vulnerabilities. Think of how should files are created in your environment and modify the directive below accordingly:

```
# Restrict access to backup files
<LocationMatch /*(\~|\.bak|\.sav|\.orig|\.old|\.20[0-9][0-9][0-1][0-9][0-3][0-9])$>
    deny from all
</LocationMatch>
```

Control those Load Modules

One key change that you want to make is to comment out load modules that are not appropriate for your environment. The default install enables all the load modules that are shipped with OHS except for userdir and php5. We cannot tell you which load modules you should comment out because it depends on your intended use. If you have a Standalone OHS 2.0 installation being used as a reverse proxy server, then you would need mod_proxy enabled. If you are using OHS 1.3 in a 10gAS configuration with a set of J2EE containers, then you probably can disable mod_proxy. You need to evaluate which ones you really need and disable all the others. It may take a little trial and error to determine your minimum level of load modules. This is a one-time effort because the typical environment does not change that significantly once you establish your baseline. As an example, we do not use php in the environment. We disable all php load modules for all OHS configurations, since we don't use this technology. We have filtering rules on the reverse proxy to filter out php requests. We still take the step to disable php on the application tier even though none of these requests should get through. Remember the defense in depth strategy. If someone was able to get past the php filter on the web tier, the restriction on the application tier OHS would still block execution of php.

There is one thing you should be aware of when doing subsequent patching of the OHS setup. A patch might re-enable a module that you disabled. You should always recheck the hardening steps to verify that they are still in place following the application of a patch. In some cases, you might have to restore the configuration prior to applying a patch because the patch may be surprised by some of the hardening steps that you have taken. You always want to keep a copy of the original configuration file in case you need to revert to the default setup. You should backup any custom configuration changes that you made to the OHS setup prior to applying a patch to prevent any of your changes from being overwritten.

The following table provides some samples of load modules that could be disabled. This list was established based on using OHS 2.0 as a reverse proxy server and using mod_security.

Table 2 - Sample configuration changes to limit load modules

Original Configuration	Revised Configuration
`LoadModule file_cache_module`	`#####LoadModule file_cache_module`
`LoadModule vhost_alias_module`	`#####LoadModule vhost_alias_module`
`LoadModule env_module`	`LoadModule env_module`
`LoadModule log_config_module`	`LoadModule log_config_module`
`LoadModule mime_magic_module`	`#####LoadModule mime_magic_module`
`LoadModule mime_module`	`LoadModule mime_module`
`LoadModule negotiation_module`	`LoadModule negotiation_module`
`LoadModule include_module`	`#####LoadModule include_module`
`LoadModule autoindex_module`	`#####LoadModule autoindex_module`
`LoadModule dir_module`	`#####LoadModule dir_module`
`LoadModule cgi_module`	`#####LoadModule cgi_module`
`LoadModule cgid_module`	`#####LoadModule cgid_module`
`LoadModule asis_module`	`#####LoadModule asis_module`
`LoadModule imap_module`	`#####LoadModule imap_module`
`LoadModule actions_module`	`#####LoadModule actions_module`
`LoadModule speling_module`	`#####LoadModule speling_module`
`#LoadModule userdir_module`	`#####LoadModule userdir_module`
`LoadModule alias_module`	`LoadModule alias_module`
`LoadModule access_module`	`LoadModule access_module`
`LoadModule auth_module`	`#####LoadModule auth_module`
`LoadModule auth_anon_module`	`#####LoadModule auth_anon_module`
`LoadModule auth_dbm_module`	`#####LoadModule auth_dbm_module`
`LoadModule cern_meta_module`	`#####LoadModule cern_meta_module`
`LoadModule expires_module`	`LoadModule expires_module`
`LoadModule headers_module`	`LoadModule headers_module`
`LoadModule usertrack_module`	`LoadModule usertrack_module`
`LoadModule unique_id_module`	`#####LoadModule unique_id_module`
`LoadModule setenvif_module`	`LoadModule setenvif_module`
`LoadModule fastcgi_module`	`#####LoadModule fastcgi_module`
`LoadModule perl_module`	`#####LoadModule perl_module`
`LoadModule php4_module`	`#####LoadModule php4_module`
`#LoadModule php5_module`	`#####LoadModule php5_module`
`LoadModule ossl_module`	`LoadModule ossl_module`
`LoadModule rewrite_module`	`LoadModule rewrite_module`

Original Configuration	Revised Configuration
	`LoadModule security_module`
`LoadModule proxy_module`	`<IfDefine RUNPROXY>`
	`      LoadModule proxy_module`
	`      LoadModule proxy_http_module`
	`      LoadModule proxy_connect_module`
	`</IfDefine>`
	`LoadModule certheaders_module`
	`<IfDefine TESTENV>`
`LoadModule status_module`	`      LoadModule status_module`
`LoadModule info_module`	`      LoadModule info_module`
	`</IfDefine>`

You can see several places where we have used the IfDefine technique to control load module loading.

WEEK

PHASE 2: ADVANCED OHS HARDENING AND VERIFICATION

FINGERPRINTING OHS

Fingerprinting the web server is a technique to identify the details of a web server configuration. If the web server configuration is known, then an attacker can select the known vulnerabilities for that configuration and immediately probe for an opening with the most likely exploits. Hiding the details of the configuration means that attacks must use a wide range of probes to find vulnerabilities. This provides you with additional time to detect the attack and respond. It's a red flag when you start seeing requests come in for technology that you do not use. We have had "friendly" scans done against the environment and received false positive results that we were vulnerable to an IIS exploit. The scanner had incorrectly identified the web server, and thought we were running an old version of IIS with a known vulnerability.

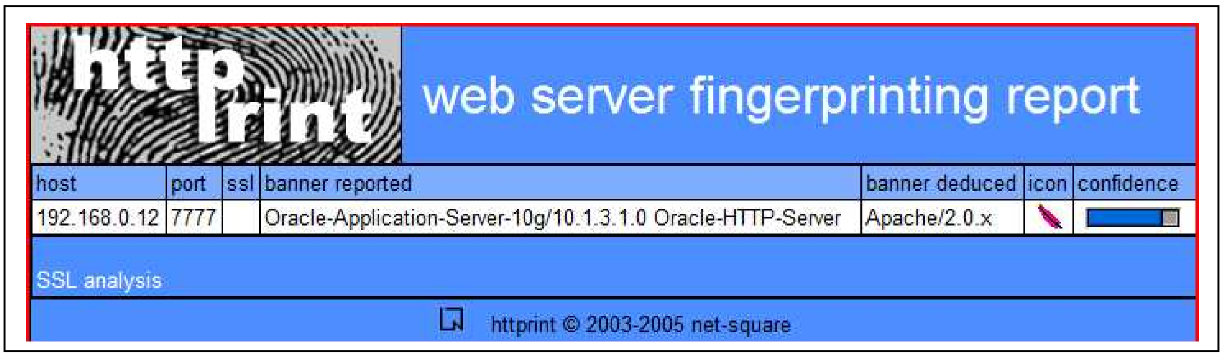
There are a number of places where information is revealed about the server configuration. The obvious place is the HTTP Header listing the web server version and turning this off is one of a number of steps you should take. However, there are other places, such as error pages, where configuration information will be displayed unless you modify the default setup. The fingerprinting techniques have become much more sophisticated with scanners detecting subtle differences in the way a particular web server responds to a bad request, the order of HTTP headers, and the format of certain responses. Closing these information leaks is one of the first steps you can take.

To start this process we can use a tool, called HTTPrint (http://www.net-square.com/httprint/) to fingerprint the default installation. We will then modify the server configuration and obscure the information leaks to see if we can get this tool to mis-identify the web server.

We downloaded the tool and ran it against a fresh install of OHS 2.0 Standalone. The tool launches a number of tests against the site to see how it responds and compares the results to a set of signatures for a variety of web servers. Here is a list of some of the requests that the tool sends:

```
192.168.0.10 - - [01/Mar/2009:11:24:09 -0500] "PUT / HTTP/1.0" 403 160
192.168.0.10 - - [01/Mar/2009:11:24:09 -0500] "JUNKMETHOD / HTTP/1.0" 403 160
192.168.0.10 - - [01/Mar/2009:11:24:09 -0500] "GET / JUNK/1.0" 403 160
192.168.0.10 - - [01/Mar/2009:11:24:09 -0500] "get / http/1.0" 403 160
```

The tool produces a report with a confidence rating to indicate which web server is being run. While looking at the server header is one check, the tool takes into account that the server header could have been modified to another value or turned off. The header is only one of many checks being done. If you look at the types of requests, you will see invalid methods and protocols being used to detect the type of response to each case. The report that was generated from the baseline run shows Apache 2.0 with a high degree of confidence.


httPrint
web server fingerprinting report

host	port	ssl	banner reported	banner deduced	icon	confidence
192.168.0.12	7777		Oracle-Application-Server-10g/10.1.3.1.0 Oracle-HTTP-Server	Apache/2.0.x		

SSL analysis
httprint © 2003-2005 net-square

If we examine the HTTP headers and default error pages we see that OHS is displaying a number of configuration details.

```
Basic Header:
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Date: Mon, 23 Feb 2009 02:19:58 GMT
Server: Oracle-Application-Server-10g/10.1.3.1.0 Oracle-HTTP-Server
Error Page:
```

```
<body>
<h1>Not Found</h1>
<p>The requested URL /notfound was not found on this server.</p>
<hr>
<address>Oracle-Application-Server-10g/10.1.3.1.0 Oracle-HTTP-Server Server at bjm-desktop
Port 80</address>
</body>
```

CIS Apache Benchmark

The Center for Internet Security (CIS) (see reference section) produces a number of benchmarking tools to verify configurations against standard industry best practices. We use the Apache benchmarking tool to evaluate the setup we have and identify any variations. The tool works by running a check against the configuration files. It does not probe the actual web site. We end up using both kinds of tools. The configuration benchmarking tool is a first step to producing a hardened configuration. Once we have the basic setup in place we run scanning tools that perform actual tests against the site. Using both sets of tools provides better coverage to verify that we have addressed known vulnerabilities.

Let's start the process by running the CIS benchmark against the base OHS 2 install and see what we get.

```
#=========[ CIS Apache Benchmark Scoring Tool 2.10 ]==========#
[Section 1.14] Web Server Software Obfuscation General Directives
 [FAILED]      ServerSignature is "On"
[Section 1.18] Access Control Directives
 [PASSED]      Directory entry for "/" is properly configured. allowoverride None
 [FAILED]      Directory entry for "/" is not properly configured. options FollowSymLinks
 [FAILED]      Directive "deny" Directory entry for "/" is not defined.
[Section 1.20] Directory Functionality/Features Directives
 [FAILED]      Did not disable Option directive "Includes" for DocumentRoot
[Section 1.21] Limiting HTTP Request Methods
 [FAILED]      There is no LimitExcept directive for DocumentRoot
[Section 1.23] Remove Default/Unneeded Apache Files
 [VERIFY]      Verify DocumentRoot files are not default Apache files.
…
 [Apache Benchmark Score]:  2.79 out of 10.00]
```

The overall score is not that good.

After making the base changes to the OHS configuration file the revised CIS Apache benchmark score is:

```
[Apache Benchmark Score]:  8.14 out of 10.00]
```

We hope you were not expecting a perfect 10! This is the real world where perfect scores are not attainable. What we were looking to achieve was to improve the configuration by hardening it, and explain any differences between the benchmark checks and the configuration we implemented. Let's examine some of the categories to explain the variances.

The CIS benchmark checks for Apache load modules that are compiled into the binary. We use an Oracle provided version, so we cannot control the modules that are compiled into it. This means certain checks will not pass:

```
[Section 1.9] Configure the Apache Software
 [FAILED]      Unless required, module "mod_status" should not be compiled into Apache.
```

For performance reasons we turn off hostname lookups. We can post process the access log when generating end user reports to retrieve the hostname.

```
[Section 1.11] Server Oriented General Directives
 [FAILED]      HostnameLookups is off for Apache Web Server
```

We have application transactions that take more than one minute to execute. This includes Discoverer report queries and EBS transactions. We have to keep the transaction timeout at a higher value and therefore we do not meet this recommended standard.

```
[Section 1.13] Denial of Service (DoS) Protective General Directives
 [FAILED]      TimeOut value "300" is greater than the recommended "60"
```

There are a number of directory permission checks in the benchmark. Some of these recommend that an OHS directory be owned by root, while others alter the permissions to read only. We have found that this can cause issues with operations managing the instance because we limit access to the root user. We believe the multiple user setup discussed earlier in the paper provides a sufficient level of separation. We relocate log files outside of the Oracle home directory, since these files can grow substantially in size. We do nightly rotation of logs and manage log files with cron jobs to compress and remove old log files. This means we do not pass these checks in the benchmarks.

```
[Section 1.24] Update Ownership and Permissions for Enhanced Security
 [FAILED]       Owner of Log directory should be root.
```

Overall we are satisfied with the results we reached and the fact that we have measured the setup we are using against an independent benchmark. We recommend that you just don't take the defaults and instead make conscious choices about how to setup your OHS configuration. You may find that some of the techniques in this paper are applicable to your environment and you can use them "as is". In other cases you may decide to use these items as a starting point and then tailor the setup to something that is a better fit for your requirements.

If you follow all the techniques in this paper, and someone asks if your OHS environment is secure, you can confidently answer NO! Any system that is exposed to the Internet has some vulnerabilities. What you can say is that you have done appropriate due diligence to harden your OHS environment and minimize security risks.

Taking Web Server hardening to the next level

Now that you have performed basic OHS hardening, it's time to consider the more advanced options of *Reverse Proxy* and *mod_security*.

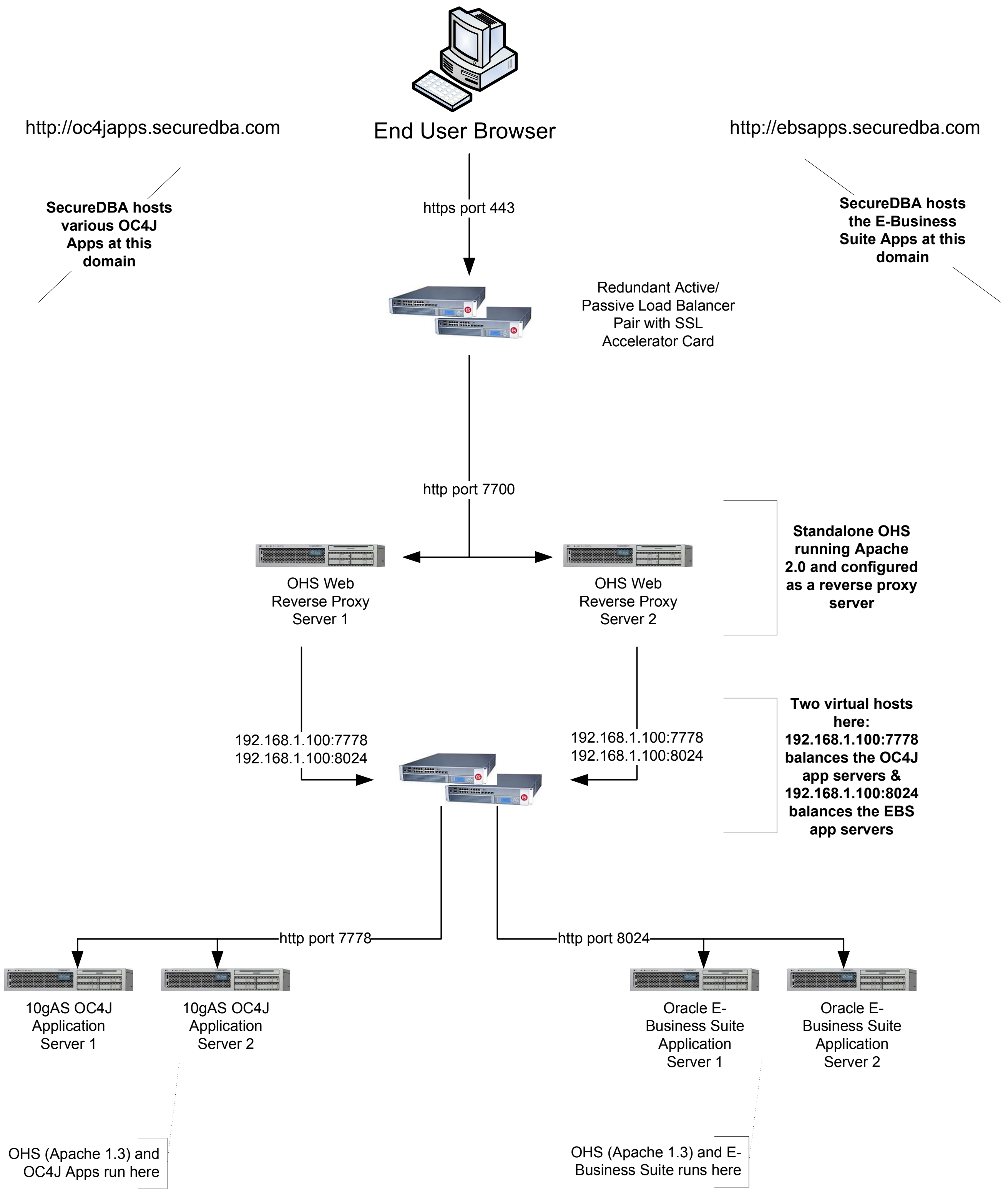
Configuring OHS as a Reverse Proxy Server

First of all, just what is a reverse proxy server and why should you consider one. For our purposes, a reverse proxy server is an instance of OHS that:

- takes an inbound HTTP request and forwards it to your web servers thus providing a layer of obfuscation;
- based on rules you define, either passes (proxies) a request onward or denies it access and therefore you can configure if to limit probes be individuals trying to fingerprint your environment;
- can serve up static content to take some load off of your web/application servers;
- can act as a server-side cache; and
- can compress content

For the purposes of this paper, we are only going to cover the first two bullets.

EXAMPLE ENVIRONMENT


http://oc4japps.securedba.com
End User Browser
http://ebsapps.securedba.com
SecureDBA hosts various OC4J Apps at this domain
https port 443
SecureDBA hosts the E-Business Suite Apps at this domain
Redundant Active/ Passive Load Balancer Pair with SSL Accelerator Card
http port 7700
Standalone OHS running Apache 2.0 and configured as a reverse proxy server
OHS Web Reverse Proxy Server 1
OHS Web Reverse Proxy Server 2
192.168.1.100:7778
192.168.1.100:8024
192.168.1.100:7778
192.168.1.100:8024
Two virtual hosts here: 192.168.1.100:7778 balances the OC4J app servers & 192.168.1.100:8024 balances the EBS app servers
http port 7778
http port 8024
10gAS OC4J Application Server 1
10gAS OC4J Application Server 2
Oracle E-Business Suite Application Server 1
Oracle E-Business Suite Application Server 2
OHS (Apache 1.3) and OC4J Apps run here
OHS (Apache 1.3) and E-Business Suite runs here

Assume that SecureDBA, LLC hosts both OC4J applications and E-Business Suite Applications on separate pools of application servers both fronted by the same pair of OHS Reverse Proxy Servers. Our goal is to proxy valid traffic forward to the correct pool of application servers and deny any traffic that does not represent a uri hosted by SecureDBA's applications. There are two *Virtual IPs* that live on the second pair (application tier) of load balancers:

- 192.168.1.100:7778 traffic will be forwarded to the OC4J application server pool by the load balancer
- 192.168.1.100:8024 traffic will be forwarded to the EBS application server pool by the load balancer

Therefore, our job is to setup the reverse proxy server to pass OC4J traffic to 192.168.1.100:7778 and EBS traffic to 192.168.1.100:8024 and deny all other traffic. The load balancer will take care of the rest.

VIRTUAL HOST SETUP IN HTTPD.CONF

It is not really required but it is a best practice to set up virtual hosts in OHS for every website you host. This allows you to have separate configurations within OHS that are specific to that host. We'll be using different reverse proxy directives for these two websites so we'll setup virtual hosts in the reverse proxy server. The pertinent snippets of the httpd.conf configuration would look something like this:

```
…
Listen 7700
…
NameVirtualHost *:7700
include "/ORACLE_HOME/ohs/conf/secureDBA_virtualhost.conf"
```

Note that we put very little detail in the httpd.conf file. Keep it simple and put your environment specific details in include files instead. Next we look at the secureDBA_virtualhost.conf file which defines the virtual host for the two sites supported by SecureDBA:

```
### This is the VirtualHost file that includes for SecureDBA's sites ###
<VirtualHost *:7700>
    ServerName oc4japps.securedba.com:80
    <Proxy *>
       Order deny,allow
       Deny from all
       Allow from all
    </Proxy>
    include "/ORACLE_HOME/ohs/conf/oc4j_proxy.conf"
</VirtualHost>

<VirtualHost *:7700>
    ServerName ebsapps.securedba.com:80
    <Proxy *>
       Order deny,allow
       Deny from all
       Allow from all
    </Proxy>
    include "/ORACLE_HOME/ohs/conf/ebs_proxy.conf"
</VirtualHost>
```

So here we see our two virtual hosts. The *Proxy ** is simply a container for the directives applied to this particular proxy. This one is not particularly interesting as we are allowing everyone access to everything. You can add more granular access control by replacing the * with a location (such as /my_uri) and then allow only specific IPs or subnets to proxy to that location.

Finally, we'll look at the two *.proxy.conf files which are at the heart of the reverse proxy configuration.

REVERSE PROXY SETUP

Let's first take a look at oc4j_proxy.conf:

```
### This is for proxy config for OC4J ###
<IfModule mod_proxy.c>
    ProxyRequests Off
    ProxyPreserveHost Off
    ProxyTimeout 300

ProxyPass /oc4j_mountpoint_1 http://192.168.1.100:7778/oc4j_mountpoint_1
ProxyPassReverse /oc4j_mountpoint_1 http://192.168.1.100:7778/oc4j_mountpoint_1

ProxyPass /oc4j_mountpoint_2 http://192.168.1.100:7778/oc4j_mountpoint_2
ProxyPassReverse /oj4j_mountpoint_2 http://192.168.1.100:7778/oc4j_mountpoint_2
</IfModule>
```

The *ProxyRequests* directive is critical. If set to *On*, it allows your reverse proxy server to act as a forward proxy. Given that we have not blocked any traffic and assuming this is an internet-hosted site, turning *ProxyRequests On* means you are now an open forward proxy server on the internet. Needless to say this is very bad from both a performance impact on your server as well as a security issue for the internet at-large. The default is *Off*, but we like to explicitly set it anyway.

ProxyPreserveHost Off means to pass the hostname the ProxyPass directive (192.168.1.100) to the backend server and not the hostname from the original request (oc4japps.securedba.com). If you are using name-based virtual hosts on the application tier then you should turn this on.

ProxyTimeout 300 will gracefully end requests that take longer than 300 seconds to get a response from the application server. 300 is the default.

The *ProxyPass* directive is used to proxy requests for a specific location to a backend server. Here the locations are simply two OC4J containers and the backend server is the load balanced VIP for the OC4J application servers.

ProxyPassReverse is needed to ensure that self-referencing redirects on the application tier do not bypass the reverse proxy. That is, with this directive, a redirect on the application server to http://192.168.1.100/oc4j_mountpoint_2/image1.gif will get redirected to http://oc4japps.securedba.com/oc4j_mountpoint_2/image1.gif. Without this directive, the redirect will bypass the reverse proxy and go directly to http://192.168.1.100/oc4j_mountpoint_2/image1.gif .

Note that requests for anything other than the two OC4J mount points will fail with a 404 since they will not be proxied and they presumably have no matching content in htdocs. This is a good way to stop probing of your site with invalid URIs.

Finally let's look at ebs_proxy.conf:

```
### This is for EBS proxy
<IfModule mod_proxy.c>
    ProxyRequests Off
    ProxyPreserveHost Off
    ProxyTimeout 300

ProxyPass /OA_MEDIA http://192.168.1.100:8024/OA_MEDIA
ProxyPassReverse /OA_MEDIA http://192.168.1.100:8024/OA_MEDIA

ProxyPass /OA_TEMP http://192.168.1.100:8024/OA_TEMP
ProxyPassReverse /OA_TEMP http://192.168.1.100:8024/OA_TEMP

ProxyPass /OA_HTML http://192.168.1.100:8024/OA_HTML
ProxyPassReverse /OA_HTML http://192.168.1.100:8024/OA_HTML

ProxyPass /OA_SECURE http://192.168.1.100:8024/OA_SECURE
ProxyPassReverse /OA_SECURE http://192.168.1.100:8024/OA_SECURE

ProxyPass /media http://192.168.1.100:8024/media
ProxyPassReverse /media http://192.168.1.100:8024/media

ProxyPass /OA_JAVA http://192.168.1.100:8024/OA_JAVA
ProxyPassReverse /OA_JAVA http://192.168.1.100:8024/OA_JAVA

ProxyPass /oa_servlets http://192.168.1.100:8024/oa_servlets
```

```
ProxyPassReverse /oa_servlets http://192.168.1.100:8024/oa_servlets

ProxyPass /servlet http://192.168.1.100:8024/servlet
ProxyPassReverse /servlet http://192.168.1.100:8024/servlet

ProxyPass /servlets http://192.168.1.100:8024/servlets
ProxyPassReverse /servlets http://192.168.1.100:8024/servlets

ProxyPass /html http://192.168.1.100:8024/html
ProxyPassReverse /html http://192.168.1.100:8024/html

ProxyPass /OA_CGI http://192.168.1.100:8024/OA_CGI
ProxyPassReverse /OA_CGI http://192.168.1.100:8024/OA_CGI

ProxyPass /CACHE http://192.168.1.100:8024/CACHE
ProxyPassReverse /CACHE http://192.168.1.100:8024/CACHE

ProxyPass /pls http://192.168.1.100:8024/pls
ProxyPassReverse /pls http://192.168.1.100:8024/pls

ProxyPass /OA_JAVA http://192.168.1.100:8024/OA_JAVA
ProxyPassReverse /OA_JAVA http://192.168.1.100:8024/OA_JAVA
</IfModule>
```

Very similar to the OC4J proxy. The only difference is now we use the other VIP listening on 8024 to get to the EBS pool and of course, the URIs here are some of those found in the E-Business Suite.

CONFIGURING OHS

For additional advanced hardening let's take a look at a real vulnerability issue …

The weather has been very windy lately. Some people in the neighborhood put out their garbage and don't put a tight lid on the trash can. Of course, the trash blows down the street and into other yards. Now a few of the people in the neighborhood do not see this as a big problem. After all their yard looks fine, and their trash is gone. The president of the homeowners association has been sending out emails to remind people to put a tight lid on their trash cans.

Software vendors or application providers sometimes have the same attitude when you report a cross site scripting or HTTP response splitting issue. The reaction seems to be "How does this affect me"?

These types of vulnerabilities do not directly affect the application that is creating the issue, but they impact the reputation of the whole community, because a person is launching an attack based on the reputation of the web site. The manager of the web server needs to operate as a community manager and police the site.

If you look at the SANS top cyber security risks for 2009 (http://www.sans.org/top-cyber-security-risks/), you will see cross-site scripting at the top of the list. The WASC Web Application security statistics (http://projects.webappsec.org/Web-Application-Security-Statistics) show HTTP response splitting near the top of the list, along with cross-site scripting and SQL injection.

The root cause of many of these vulnerabilities is due to software developers setting HTTP headers, such as cookies and redirects, from unvalidated user input. Here is an example using a vulnerability that was reported in the Apache mod_status module. A patch is available for this issue. It is considered a moderate level risk because the Apache status page should not be publicly available, and it is disabled by default. This is a link to a description of the issue: CVE-2007-6388 - Cross-site scripting (XSS) vulnerability in mod_status (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6388).

The mod_status module included a feature to refresh the screen using a user entered value for the number of seconds. The refresh parameter normally contains a number of seconds. However, the parameter allows the specification of a URL, and the browser will refresh to the URL specified. The mod_status module did not check the user input. It simply took the value of the parameter and rewrote it as an HTTP refresh header. This means a user could click a link expecting to go to a trusted site showing server-status, and instead be redirected to another site on the Internet. This is what the request/response looks like:

telnet my-trusted-site.com 80

```
Trying 127.0.0.1...
Connected my-trusted-site.com.
Escape character is '^]'.
GET /server-status?refresh=0;url=http://untrusted-site.com/ HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 13 Jan 2010 17:48:03 GMT
Refresh: 0;url=http://untrusted-site.com/
Connection: close
Content-Type: text/html
```

The value of the refresh parameter that was entered by the user, and not validated, was placed in an HTTP header in the response causing the browser to immediately redirect to a new site. A user clicking this link would think they were going to my-trusted-site, but instead would be redirected to untrusted-site.

An HTTP header is like a trash can lid. If you don't keep a tight lid on HTTP headers, then garbage will be blown all over the community.

MOD_REWRITE AND MOD_SECURITY

Two of the standard tools that we use to filter URLs are mod_rewrite and mod_security. Both of these modules are part of the standard OHS distribution, which means Oracle provides support. Oracle has a number of Metalink notes regarding use of these tools. Please see the reference section for links to both Oracle notes about these modules and general industry references.

Mod_rewrite was designed to be a tool for URL rewriting. It uses a set of rules based on regular expression matching to check the incoming request URL and perform an action that modifies the request. This provides the capability to block requests by redirecting the incoming request to an error page. Mod_security was designed to be a web application firewall tool. It uses a set of rules based on regular expression matching to check the incoming request URL and perform an action that modifies the request. There is certainly some overlap between the two tools. This summary provides a quick comparison between the two.

Table 3 - Comparing mod_security and mod_rewrite

mod_security	mod_rewrite
Pro	Pro
Availability of Rules Detailed logging Designed as a security tool	Typically already in use Good for simple blocking Performance
Con	Con
New module to maintain Parsing adds overhead OHS uses old 1.84 version	More work to code rules Logging more for debug Not designed for security

Using either tool for blocking bad requests will produce the same output to the end user. As an example, if we could setup a rule to block the PUT method then the rule would redirect the user to an error page. Below is an example of the format of the rule used for each tool to accomplish this.

Table 4 - mod_security vs mod_rewrite rule formats

mod_security	mod_rewrite

`SecFilterSelective REQUEST_METHOD "PUT"`
`"id:888000,deny,log,status:405"`
	`RewriteCond %{REQUEST_METHOD} ^PUT`
	`RewriteRule .* - [F]`

One of the primary differences between the two tools is logging. Mod_security includes more extensive logging that is useful to security personnel that are tracking and investigating incidents. Mod_rewrite uses logging more for debugging. Another significant difference is that mod_security supports a pass, log action. There are a number of gray issues when you are evaluating whether or not a request is valid. The pass, log action allows you to log the request when it matches the rule, but does not block or modify the request. This approach provides a method to evaluate a rule without impacting an application. The following table illustrates the difference in logging between the two tools.

Table 5 - mod_security vs mod_rewrite logging

mod_security	mod_rewrite
`== UNIQUE_ID: QBywrH8AAQEAAHrxXvwAAAAA Request: 127.0.0.1 - - [12/Mar/2009:20:33:32 --0400] "PUT /test HTTP/1.0" 405 339 Handler: (null) -- PUT /test HTTP/1.0 mod_security-message: Access denied with code 405. Pattern match "PUT" at REQUEST_METHOD mod_security-action: 405 HTTP/1.0 405 Method Not Allowed Allow: TRACE Content-Length: 339 Connection: close Content-Type: text/html; charset=iso-8859-1`	`127.0.0.1 - - [12/Mar/2009:20:38:12 --0400] [bjm-desktop/sid#8fe7e88][rid#95d97c8/initial] (2) init rewrite engine with requested uri /test` `127.0.0.1 - - [12/Mar/2009:20:38:12 --0400] [bjm-desktop/sid#8fe7e88][rid#95d97c8/initial] (3) applying pattern '.*' to uri '/test'` `127.0.0.1 - - [12/Mar/2009:20:38:12 --0400] [bjm-desktop/sid#8fe7e88][rid#95d97c8/initial] (4) RewriteCond: input='PUT' pattern='^PUT' => matched` `127.0.0.1 - - [12/Mar/2009:20:38:12 --0400] [bjm-desktop/sid#8fe7e88][rid#95d97c8/initial] (2) forcing '/test' to be forbidden`

We initially started using mod_rewrite rules to block obviously invalid request URLs. As we evolved and needed more rules and the ability to deal with these gray issues we implemented mod_security. We use both tools, with mod_rewrite handling the chore of rewriting request URLs and mod_security handling filtering. If all you need is basic blocking of invalid requests, you may find that mod_rewrite is sufficient. Anyone running the E-Business suite may have noticed that Oracle distributes configuration files that use both tools. The security.conf file contains mod_security rules, while the url_fw.conf file contains mod_rewrite rules.

If you decide to use mod_security you can download free rule sets or purchase more up-to-date rule sets from vendors that provide support. It makes sense to structure your rule sets into categories to better organize the rules and make them easier to maintain. We have one rule set file containing just the list of user agents. In this file we initially test for the user agents that are typical for the site and that we recommend be used. We allow requests with these user agents to pass through and skip the remaining user agent checks. Requests that do not come from supported user agents are filtered through a list of known bots, spiders, and scanners. We block those user agents that are not allowed. The request is then evaluated by additional rule sets. We have some application specific rule set files, and files that differ between Intranet-hosted and Internet-hosted sites.

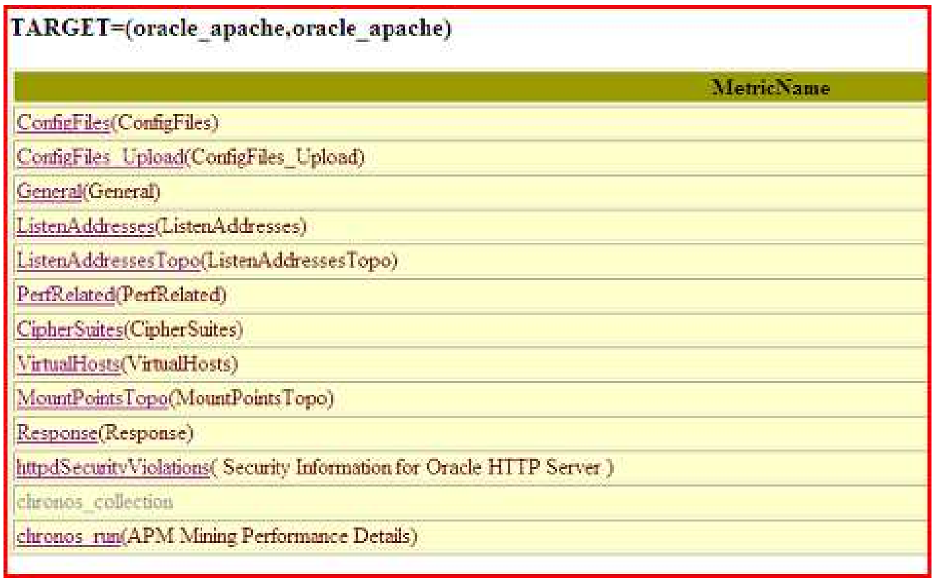
One final point concerns the HTTP error code that is returned when a request is blocked. Some of the documentation mentions returning a code 500 – Internal Server Error when blocking requests. This can be confusing for the Operations team because they associate a 500 status with a system problem and set monitoring alerts when these get triggered. We use the status code 400 – Bad Request for a majority of the mod_security rules. This indicates that the request input was unacceptable. We use this status to indicate that the request input did not pass, however if the client modifies the request it might be acceptable. If the user enters a request with the text “alter user sys identified by”, maybe it was a typo. We will still flag it and log the request, but we want to distinguish this from obviously bad requests. We use another status code to indicate that requests are always prohibited. If a user sends a request using the Trace method, then a 405 – Invalid method is returned. Other obvious errors might return a 406 – Not Acceptable. Oracle uses the 410 – Gone code for the mod_rewrite rules implemented in EBS. Some of the security documentation suggests returning just the 404 – Not Found status for all situations, so you are not revealing any additional information. Returning this status can confuse Operations and generate more support calls. We avoid using the 403 – Forbidden status in mod_security rules because this status is generated from the main configuration rules. If we get a report of a problem where a valid request is being denied, the status code helps to identify where the rule comes from and which log to check. This approach is useful when looking at web site summary statistics by error code too.

GRID CONTROL SETUP AND RECOMMENDED OHS METRICS

To close out week one we want to setup the Grid Control agent and register it. The agent includes a metric browser to check the status of the OHS install. Even though you may not have installed or configured the central OMS server you can still install the agent, discover the OHS installation, and enable the metric browser to confirm that it is working. We have found that when running the agent discovery (agentca –d), the agent may not detect the installation. This is particularly true for the standalone OHS based on Apache 2.0. If you are having difficulty discovering the installation, check the targets.xml file in the agent home. It should have entries like the ones below:

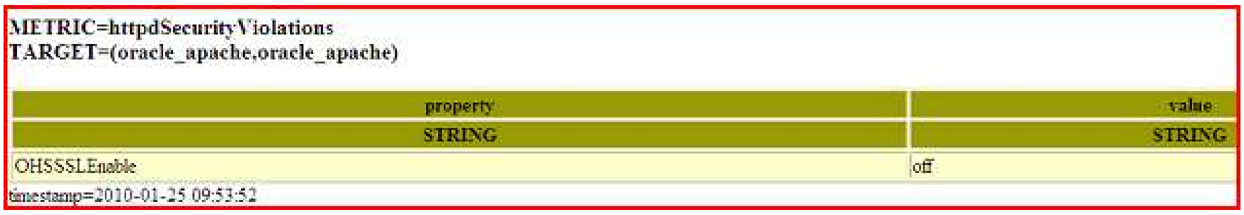
```
<Target TYPE="oracle_apache" NAME="INSTANCE.HOSTNAME">
    <Property NAME="HTTPMachine" VALUE="HOSTNAME"/>
    <Property NAME="HTTPPort" VALUE="7777"/>
    <Property NAME="version" VALUE="stdApache10.1.2"/>
    <Property NAME="OracleHome" VALUE="/u01/oracle/ohs/2.0"/>
    <Property NAME="OracleBinaryHome" VALUE="/u01/oracle/ohs/2.0"/>
    <Property NAME="OPMNPort" VALUE="6201"/>
    <Property NAME="logFileDir" VALUE="/u01/oracle/logs/ohs"/>
    <Property NAME="logFileName" VALUE="access_log"/>
</Target>
```

You can check the .trc log file to confirm the agent discovered the installation. Once that is done modify the emd.properties to enable the metric browser, and bounce the agent. You should now be able to see the OHS metrics screen.


TARGET=(oracle_apache,oracle_apache)

MetricName
ConfigFiles(ConfigFiles)
ConfigFiles_Upload(ConfigFiles_Upload)
General(General)
ListenAddresses(ListenAddresses)
ListenAddressesTopo(ListenAddressesTopo)
PerfRelated(PerfRelated)
CipherSuites(CipherSuites)
VirtualHosts(VirtualHosts)
MountPointsTopo(MountPointsTopo)
Response(Response)
httpdSecurityViolations(Security Information for Oracle HTTP Server)
chronos_collection
chronos_run(APM Mining Performance Details)

The OHS metrics includes a combination of information from the configuration file and real time requests to the OHS web site. The majority of the metrics are performance based. If you check the httpdSecurityViolations, you will find very few items being checked.



METRIC=httpdSecurityViolations
TARGET=(oracle_apache,oracle_apache)

property	value
STRING	STRING
OHSSSLEnable	off

timestamp=2010-01-25 09:53:52

MONTH

PHASE 3: MONITORING THE WEB SERVER ENVIRONMENT

GRID CONTROL POLICY OVERRIDES FOR OHS

Grid Control includes a very limited set of policies that cover OHS. If you check the counts in the MGMT$POLICIES view by target_type you will see this summary.

Table 6 - Grid Control policy count by target type

TARGET_TYPE	Policy-Count
oracle_database	145
rac_database	66
oracle_listener	36
oracle_apache	8
host	5
osm_instance	4
oracle_webcache	4
weblogic_j2eeserver	3
oc4j	2
oracle_emrep	1
oracle_calendar_ocal	1

There are only 8 policies that are specific to OHS:

Table 7 - Grid Control OHS standard policies

METRIC_NAME	DESCRIPTION	IMPACT	RECOMMENDATION
HTTPServer HostNameLookups	Check that HostNameLookup is off on this HTTP Server	Oracle has found that performance degraded by a minimum of about 3% in our tests with HostNameLookups set to on.	Oracle recommends that Host Name Lookup should be turned off.
HTTPServer MaxKeepAliveRequests	Check that MaxKeepAliveRequests directive is set to a non-zero value on this HTTP Server	If MaxKeepAliveRequests is set to zero then unlimited number of connections, Httpd server process cannot be used to service other requests until either the client disconnects, or the connection times out.	Oracle recommends that MaxKeepAliveRequests directive should not be set to zero.
HTTPServer DirectoryIndexing	Verifies that Directory Indexing is disabled	If indexing is on, a malicious user may be able to view restricted files and directories in the Document Root directory.	Turn off Directory Indexing.
HTTPServer	Verifies whether	Absence of an access log	Enable the access

AccessLogging	Access Logging is enabled	can severely cripple administrators' ability to monitor malicious attacks.	logging for HTTP Server.
HTTPServer OwnerAndSetuidBit	Verifies that the HTTPd binary is not owned by a super user	If HTTPd is owned by root and the setuid bit is set, malicious users may be able to gain access to the system as a super user.	A user other than root should own the HTTPd binary.
HTTPServer WritableFiles	Checks whether users other than the owner have write permission in the Document Root folder	Malicious users may be able to overwrite a writable file in the Document Root directory.	Do not include any group or world writable files in the Document Root folder.
OHSDummyWallet	Checks whether a Dummy Wallet is being used on HTTP Server	Use of a Dummy Wallet provided by Oracle can severely compromise the security of the site.	Do not use a Dummy Wallet for production SSL load.
HTTPServer OnSSL	Checks whether Secure Socket Layer (SSL) is enabled for Single Sign-On (SSO) on HTTP Server	If SSL is not enabled on HTTP Server, malicious users may detect the user name and password entered by a user.	For secure transmission of user name and password, enable SSL on HTTP Server

While the default policies are fine, they are insufficient to thoroughly check the web server configuration. There are many key areas that are not covered by these policies. The Grid Control policies should be augmented with additional checks. Oracle provides a method to extend Grid Control to add user specific policies. Based on the guidelines discussed in this paper the following additional policy checks should be added.

Table 8 - Grid Control Additional OHS policies

METRIC_NAME	DESCRIPTION	IMPACT	RECOMMENDATION
HTTPServerXX ServerSignature	Check that the server signature directives have been set to disable display of server information.	Revealing the server makes it easier to craft attacks	Use None to hide the server information
HTTPServerXX Version	Verify the version of OHS that is installed	Difficult to patch depending on version used	Run an application server version
HTTPServerXX NotPatched	Determine if OHS has been patched	The web server is the most exposed component	You must maintain the outer defense
HTTPServerXX ThreeUser	Three users should be setup to install, execute, and serve content	Reduce risk by creating a separation of duties.	Use the three user/group setup to minimize impact and for damage control
HTTPServerXX DisableLoadModule	Load modules not needed for this OHS function should be disabled	Only run the components that you need. Don't provide other attack surfaces unnecessarily	Use the IfDefine technique to control which modules get loaded
HTTPServerXX	Uses custom error documents to	Server information may be leaked if you use default	Create simple error docs to replace

ErrorDocs	minimize server information	pages	default pages
HTTPServerXX DenialofService	Directives set to prevent denial of service attacks	Verify that min settings and timeouts are appropriate	Change default timeout and mix/max values to match site requirements
HTTPServerXX Fingerprinting	Use of fake HTTP headers to prevent fingerprinting the web server	Default behavior can reveal the webserver and OS version	Utilize fake headers to obfuscate the default behavior and confuse attackers
HTTPServerXX DisableMethods	Use of the limit except directive to disable HTTP methods	Blocking Trace is not enough, other methods like Options can reveal data	Limit methods to GET and POST if possible
HTTPServerXX SetOptions	Directory settings for the options directive to limit access	Options directives can allow access outside the normal web root.	Explicitly set limitations on the options directive
HTTPServerXX DemoContent	Remove demo content	Distributed content especially cgi-bin programs can be compromised	Verify that demo content is removed and recheck after patches are applied
HTTPServerXX Permissions	Limits on directory permissions	Not accounting for all directories may expose information	Include a deny all directive for the top level directory

AUDITING, REPORTING AND TRENDING (ART)

LOGGING

The default access log format for OHS is the common log format. This provides a very basic level of information, which is suitable for hit counts and standard web site reporting. We used this format for a period of time, and produced end user reports from the information. A typical common log entry looks like this:

```
192.168.0.10 - - [23/Feb/2009:21:45:58 -0500] "GET /index.html HTTP/1.1" 200 14679
```

We also use the extended format which includes the referring URL and the browser type. These additional fields provide additional detail for end user reporting. They also provide detail that can be used for investigation. The user agent field frequently contains the name of a bot, spider, or scanner. Certainly hackers can modify the user agent or fake running as another agent, but many do not bother to do this. The referrer can assist in tracking the path through the site or references from another site. The extended format would look like this:

```
192.168.0.10 - - [10/Mar/2009:21:23:17 -0400] "GET /index.html HTTP/1.1" 200 14679
"http://192.168.0.12:7777/OHSDemos.htm" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
6.0; GTB5; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.5.30729; .NET CLR
3.0.30618)"
```

Several years ago, Chris Josephes wrote an article called *"Profiling LAMP Applications with Apache's Blackbox Logs"*. Chris recommended including several additional fields in the access log. This provided detail that could be used for performance tracking and assessment. We used this "blackbox" format as the basis for the logs that we use. We did make a couple of modifications to include a few extras. The original blackbox format had more of a focus on performance and we wanted to add some security related items. The applications we support use session cookies to maintain user sessions. One of the potential exploits is capturing session cookies from cross site scripting vulnerabilities. We log the user cookies in the access log, which provides access to the session keys and makes it easy to spot anomalies between IP addresses and sessions. Since we run an OHS reverse proxy, we want to record the original client IP address passed from the proxy in addition to the proxy entry point. The OHS configuration directive looks like this:

```
LogFormat "%h %l %u %t \"%r\" %>s %B \"%{Referer}i\" \"%{User-Agent}i\" \"%{X-FORWARDED-
FOR}i\" \"%{cookie}i\" %v %X %P %T" blackbox
CustomLog /u01/app/oracle/product/10.1.3.2/companionCDHome_1/ohs/logs/access_log blackbox
```

The fields used in this format maintain the same order as the common log format for the start of the file. This allows standard reporting tools to be used against the log because the tools just ignore the extra fields.

Table 9 - Blackbox Log Parameters

Log Parameter	Definition
"%h	Source IP address
%l	Identity of the client
%u	User ID from the remote_user variable
%t	Time request completed
\"%r\"	URL request line
%>s	HTTP status code
%B	Size of object in bytes sent in response
\"%{Referer}i\"	Referring URL
\"%{User- Agent}i\"	Client User Agent
\"%{X- FORWARDED- FOR}i\"	Client IP for reverse proxy
\"%{cookie}i\"	Cookies including session cookies
%v	Virtual host name
%X	Connection Status
%P	Process id and Thread id
%T"	Time to Serve Request

The access log entry for a request in this format looks like this:

```
192.168.0.10 - - [10/Mar/2009:21:23:17 -0400] "GET /index.html HTTP/1.1" 200 14679
"http://192.168.0.12:7777/OHSDemos.htm" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
6.0; GTB5; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.5.30729; .NET CLR
3.0.30618)" "10.0.0.100" "JSESSIONID=8EEEE08C4DEFF1B72F9BCCEC72B58544" bjm-desktop + 27860
0
```

This format has helped resolve problems and provided the detail we need to investigate issues. We have been able to track user activity through the site that led to errors. We have used the cookie information to spot problems with sticky sessions when the user session was incorrectly routed to the wrong app tier. We have found cases where the sessions and IP addresses did not match up. We use inexpensive off the shelf tools to analyze the access logs for end user reporting, troubleshooting, and log analysis. For end user style reporting we use WebLog Expert.

This produces an attractive output in pdf format that is easy to distribute. You can download a free lite version or purchase a full featured version. We have used a simple perl script to perform a quick analysis of current activity to see which URLs are generating 500 errors. The find_status.pl script provides a quick summary of HTTP status codes and URLs.

This is one of a number of handy perl tools that can be used for analysis. Another quick analysis can be done with grep, such as finding transactions that took more than 5 seconds by checking the time field in the last column (grep -v [012345]$

access_log). We have used Log Parser from Microsoft to perform summaries. This is a useful tool that allows SQL queries against flat files. It does not require a database instance to execute.

ARTIFICIAL IGNORANCE

Marcus Ranum applied the term artificial ignorance to the process of monitoring log files. You build a filter of events to ignore, and then look at everything else. All the items that you consider normal get filtered out. Do you remember the Sherlock Holmes quote "*Once you eliminate the impossible, whatever remains, no matter how improbable, must be the truth.*" - Arthur Conan Doyle. It's impossible to identify all the potential attacks that might be launched against a web site. New attacks surface every day. While part of the strategy should be to look for specific types of attacks, an equally important strategy is to look for the unusual.

One of the web sites we support is frequently bookmarked by end users. When a user bookmarks a site, the browser sends a request to retrieve the favicon.ico file from the web server. This is the image that will be used for the bookmark. We currently do not have a bookmark image for the web site. This means a 404 Not Found error is returned to the browser. The user never sees this error, the browser simply continues to bookmark the site using a generic image. Web site reports always show the file favicon.ico as the top Not Found document. In our situation, this is normal. We considered modifying the response for this request to eliminate the error, but the 404 error actually serves a purpose. If we review the web site reports, we always expect to see this file with the most 404 errors. If this file is not at the top, then something unusual happened. This is a clue to do more investigation. It may mean that a popular document has been removed, either by mistake or in an attempt to deface the site.

We keep a close watch on HTTP 500 Internal Server errors because they always indicate some problem on the site. Unfortunately, some of the products that we run trap errors and return a 200 Successful code to the browser, which does not indicate any error in the logs. The message on the screen back to the user might say something like "fatal error", but the logs don't show this. One way to look for these situations is to inspect the number of bytes sent. The number of bytes for a good result frequently falls in one range, while the fatal errors tend to fall in a different range. This has helped us spot problems when all the other logs did not contain any errors.

It's important to look at other errors even if the volume is very low. If you see requests for technology that you do not support or types of content that you do not serve, then these are red flags. In our case we do not use php on the site. A request for a php file is a strong indication of some unusual condition. We block these requests and return an error code. Even though you may only see one or two requests that fall in this category it is important to investigate these further because they may mean that someone is probing the site. As Holmes said "*You know my method. It is founded upon the observation of trifles.*" Use this method when setting up your log analysis.

KEY METRICS FOR MONITORING THE WEB SERVER

Grid Control and the Apache status module provide plenty of performance metrics to determine if the web server setup is able to handle the web site load. We assume that you are already running reports from the OHS access logs using one of the many commercial or free web log analysis tools. These tools will provide plenty of web site reports on frequently accessed pages, web site visitors, and access times. The one area that is not well covered with these tools is security. If you have enabled mod_security, then you will have an audit log which contains the requests that triggered mod_security rules. Summarizing the mod_security-message that appears in the log will provide a quick indication of the rules that are catching requests. In some cases these will be false positives, and you will need to modify the mod_security rules to eliminate these issues. The remaining cases will need more investigation. One of the benefits of mod_security is the detail contained in the audit log. Whether or not you have enabled mod_security you should not use this as the only step in the analysis. The OHS error_log and access_log include additional information that should be reviewed.

There are at least two items we look for when examining the error_log. One situation we look for is segmentation faults. This means the OHS process that was handling the request core dumped. This will almost always only effect the individual request, and not the entire web server process. This means if you look at the process uptime in opmnctl you will not see any issue. If the process handling the request has a segmentation fault, then entries will not be written to the other logs. The fault could be due to a bad request that a client is purposely sending. You may need to turn op the LogLevel in the OHS config to debug to capture more details about the error in order to open an SR with Oracle. Another situation to look for is unexpected restarts of the web server. OHS can be restarted gracefully, which means requests will be completed and then the parent process will reread the configuration files and pick up any changes. This process does not impact the uptime of the process. This is a useful feature if you are the one making the configuration change and you want to implement the change without causing any downtime. This feature could also be used by someone who is attempting to change the web server configuration and needs to restart the process to pick up a modified configuration without triggering any alerts.

The OHS access_log can be analyzed in different ways to spot potential problems. If you have modified the custom log to use the blackbox format discussed in the previous section, then you have even more to work with. One area to focus on is what you are restricting and summarizing activity against this list. You will likely have restrictions on file types and HTTP methods using LocationMatch and LimitExcept directives. You should create a summary by restriction with a count by HTTP status code. Ideally the report you produce should be empty, which means there were no requests for restricted content. You will probably find at least a few entries on the report. We frequently find desktop products that send HTTP Options requests to the web server, and we block these requests. If you find entries on the report you should examine the HTTP status codes to make sure the counts show codes in the 4XX range such as 403-forbidden or 404-not found. Of course if you find a 200 status, then you want to go back and inspect your web server configuration to see if you are allowing invalid requests through. We have found cases like this when the directive we used was not properly applied to a virtualhost entry. You want to look at the volume of errors and the number of unique cases. If you see entries for lots of different cases it probably means someone has scanned your site looking for vulnerabilities.

One of the more common vulnerabilities is session hijacking. This can happen when someone intercepts a valid user session cookie, and uses that cookie to assume the identity of the user. To help combat this problem vendors have added options to require cookies be sent via SSL to prevent snooping. Oracle has added an OssoSecureCookies directive to single signon to transmit the single signon cookie only in secure mode. If you are logging cookies in the OHS access log, then you can run a report to look for the possibility of session hijacking. Create a report to summarize the JSESSIONID cookie or the SSO cookie by IP address. What you are looking for is different IP addresses using the same session cookie. Session cookies should not be reused. If you see two different client IP addresses with the same session cookie you need to investigate. There may be cases where businesses have multiple proxy servers that are handling outbound traffic. In those situations you might see different IP addresses for the same session id. You should be able to rule these out by eliminating cases based on the first two octets of the IP.

TIPS AND TRICKS FOR WORKING WITH OHS

Oracle has a Metalink note entitled *"Everything You Wanted to Know About the Apache-Based OHS Version"* (Doc ID: 260449.1). Well there are a lot of things about OHS that are not contained in this Metalink note that you need to know. Many of the items are contained in other Metalink notes. Unless you happen to be looking specifically for these items you might overlook a lot of good information. In this section of the paper we cover tips and tricks for managing OHS. There are lots of fun facts, and some not so fun, that you should know to manage your OHS installation. There is a summary of Metalink notes at the end of this section, which we recommend you read when planning your OHS deployment.

THE BEST FEATURE OF OHS BASED ON APACHE 2 IS NOT ENABLED BY DEFAULT!

The best feature of OHS based on Apache 2 for Unix platforms is the threaded implementation called MPM Worker. The OHS 1.3 version uses a process based model for handling each incoming request. This is a heavyweight design that uses a large footprint on the server. The process based design is not optimal for today's modern multi-core chip architectures that are optimized for threaded processing. The Apache 2 design allows either the threaded version or the process version to be used. Some applications may require the process based implementation because of plug-ins that are not thread safe. The version of OHS 2 that ships is configured by default to use processes not threads. It's an easy change to modify OPMN settings to enable the MPM Worker configuration. The Metalink note "How To Configure Worker MPM In HTTP Server Based On Apache 2.0" (Doc ID: 299125.1) shows the steps to follow.

BUILD YOUR OWN MOAT

A common defensive strategy is to build an outer perimeter to ward off attacks. This first line of defense can halt the most common attacks and protect the vital infrastructure. One way to do this with web architecture is to setup a reverse proxy. This provides a central point to filter all incoming traffic and block any suspicious traffic. Two of the most common web attacks use Cross Site Scripting and SQL Injection to exploit vulnerabilities in applications. We have found that a lot of software does not have sufficient input validation routines to protect against these attacks. It is not always an option to wait for fixes to the code, particularly if this is a COTS product. An effective strategy to defend this insecure code is to setup a reverse proxy with mod_security and/or mod_rewrite rules enabled. This provides a way to block bad requests at the outermost point, and does not require code changes from the existing applications. The one downside is that you could potentially block valid activity, which means you will need to fine tune any rules that you implement to meet your specific environment. You may have two sets of rules, one for intranet and another for internet, if you want to allow intranet users additional capabilities that you do not want to expose on the internet.

LISTEN UP!

The E-Business Suite has used a design pattern called the two listener strategy for a long time. This strategy is documented in the Metalink note *"Dual OHS Listener Configuration for Mid-Tier PL/SQL in E-Business Suite 11i"* (Doc ID: 239176.1). This approach addresses the performance issue with the process version of Apache 1.3. The standard 10gAS installation includes HTTP processes that listen on the main OHS port, 7778, and additional ports for SSL, 4443, and DMS, 7200. When hardening the OHS configuration you need to be aware of all open ports and confirm that the directives you are adding apply to the entire configuration. You may need to repeat directives in different sections of the configuration to insure that the rules are applied to each listening port.

USE AN INCLUSIVE OHS CONFIGURATION

OHS supports the use of the include directive to incorporate additional configuration directives into the main configuration. A number of the Oracle products have a separate conf file with OHS configuration directives. You should use the same technique to incorporate any local rules into the configuration. Creating rules in different configuration files keeps your rules separate from the Oracle configuration rules making upgrades and patching easier. You can also create environment specific configuration files that isolate difference between dev, test, and production environments, such as IP addresses. Well organized conf files are easier to manage. Wrap the rules you create in a set of IfModule directives to test for the presence of a load module. There are syntax differences between OHS 1.3 and 2.0 for various modules. One example is mod_headers, which has additional options that are supported in version 2. Rules that are specific to a particular version can be isolated into separate conf files to allow a common compatible set of rules to be shared by both versions reducing overall maintenance.

Can you use mod_plsql with the OHS version based on Apache 2.0?

Yes, you can. The Metalink note *"Is it Possible To Configure Mod_plsql With Apache 2.0?"* (Doc ID: 428391.1) has been updated with information about the 11g release of OHS and mod_plsql support. If you run Apex applications and are required to run mod_plsql, then you should consider this version. The threaded version of Apache 2 means a true connection pool can be used with mod_plsql. A test was run using the Apache Bench (ab) component that ships with OHS. This performs a load test by simulating end user transactions. The results were compared with a normal load of Apex users. The OHS version based on Apache 1.3 used hundreds of database connections and had a substantial impact on memory consumption on the database. The OHS version based on Apache 2.0 and configured to use the threaded MPM Worker configuration used two database connections. The major Oracle products are discontinuing use of mod_plsql, such as EBS 12 (see Doc ID: 726711.1). The mod_plsql design does not fit in well with modern web architecture and is difficult to secure. Maybe in the future Oracle will produce a servlet version of mod_plsql similar to the forms servlet that will improve the Apex product.

Which module should I use, Mod_Security or Mod_Rewrite?

Why not use both. The Metalink note *"DMZ Configuration with Oracle E-Business Suite 11i"* (Doc ID: 287176.1) suggests this approach:

1. mod_security - Reject obviously bad requests before anything else happens
2. mod_rewrite - Check for allowed URL before mod_proxy hands the request over to the external web tier
3. mod_proxy - Only proxy request that seem valid (have passed the 2 above filtering steps) to the external web tier

These modules were designed with different purposes in mind. The combination of both tools provides a rich set of capabilities that you can leverage. Oracle shows examples using both tools, and EBS distributes configuration files with various rules that apply to mod_rewrite and mod_security. It's not really important which module you pick. The key thing is to deploy rules that properly protect your environment. Use the tool that you are most comfortable using and that best fits your environment.

Using OHS based on Apache 2 with the latest Oracle products.

Oracle has acquired a large number of new products the last few years. The products that included integration with Apache are based primarily on Apache 2, not Apache 1.3. While Oracle stayed with Apache 1.3 the industry has moved on to Apache 2.0 and Apache 2.2. In some cases the products will not work, or will perform poorly with OHS based on Apache 1.3. We learned this lesson the hard way when deploying the Universal Content Management product based on the Stellent product. We attempted to set it up with OHS based on Apache 1.3 and it did not work. Of course, before attempting this we verified in Metalink that this was a certified configuration. We entered an SR and found out from the UCM developers that they just assumed OHS would work, since it was based on Apache, and never actually tested it. This was a new concept because in the past the definition of certified configuration meant that Oracle had tested it. If you plan to use one of the newer Oracle products you should consider using OHS based on Apache 2.

A bit of nostalgia

Many of the Apache load modules deployed with OHS are based on much older versions of the modules. As an example, the mod_security module that is packaged with all versions of OHS is version 1.84, which was released in August 2004. The latest version of mod_security is 2.5. The format of the rule sets for mod_security changed with version 1.9 and changed again with version 2.0. There are a number of widely available rule sets for mod_security that you can download and adapt to your environment. However, many of these rules are based on the later versions of mod_security. This means it takes additional effort to convert rules designed for version 2 to the 1.8 format.

Virtualization

OHS has its own virtualization technique using the Apache virtualhosts directive. Even a basic 10gAS installation sets up a virtual host for SSL configuration. A common topology used for large scale web deployments is to terminate SSL on the load balancer and use hardware acceleration to speed up SSL performance. Even with this setup you may still use SSL in the 10gAS configuration to secure the administrative portion of the site, AS Console. Anytime you use virtual host configurations you need to make sure the security rules apply to each virtual host. The rules for load modules vary when supporting virtual host directives. You may need to explicitly set an inherit rule to make sure that rules loaded in the main configuration apply.

LOAD MODULE ORDER IS IMPORTANT

The order of Apache load modules is important and will affect the behavior of the configuration rules that you implement. This is particularly true in OHS 1.3. The modules are executed in the reverse order to the sequence listed in httpd.conf. You may experience issues when using combinations of mod_rewrite and modules like mod_oc4j. Simply changing the sequence of the LoadModule directives may fix the issue, see Doc ID: 403585.1. This is not as much an issue with later versions of OHS based on Apache 2 because Apache provided an option for modules to declare the phase when they should be invoked. You may still see issues when you mix a variety of modules together including mod_proxy, mod_security, and mod_rewrite. If you are not getting the results you expect, check the module sequence to see if that is an issue. Setting the OHS LogLevel to debug may help or setting explicit debug options for these modules can be helpful in debugging virtual host setups and URL rewrite rules.

TEST THOSE CHANGES

The techniques outlined in this paper require a number of changes to the OHS configuration. You may find it easier to modify the OHS configuration files directly rather than using Application Server Control to modify the entries. We typically make a change and then use the apachctl configtest option to verify the configuration before enabling it. Just ignore the warning you get when using this tool: *"WARNING!! Direct use of apachectl within Oracle9iAS is deprecated."* The option to test the Apache configuration is not part of OPMN. If the configuration change is valid, you will get a message: "Syntax OK". Once you have confirmed the syntax you can execute a graceful restart of the OHS process with the command opmnctl restartproc type=ohs. The graceful restart allows existing requests to complete, and then reads the updated configuration to pick up the changes. Remember just because the syntax is valid does not mean the change you made will work. To test the change after restarting OHS you can execute a simple telnet test simulating a browser request, see below. If you are using 10gAS 10.1.2 don't forget to execute dcmctl updateconfig after modifying any OHS configuration files.

```
      telnet localhost 7777
      Trying 127.0.0.1...
      Connected to localhost.
      Escape character is '^]'.
      TRACE / HTTP/1.0
HTTP/1.1 403 Forbidden
```

NEED A LITTLE CACHE?

Whether concerned about privacy, performance or both, web site administrators have several means at their disposal to manipulate client-side caching behavior. In particular, there are two Apache modules, mod_expires and mod_headers that allow administrators to modify HTTP response headers in order to control browser caching. Let's walk through some examples showing coarse and fine-grain control.

MOD_EXPIRES MODULE

This Apache module controls the generation of Expires headers which tell browsers how long a document should be cached locally based either on access or modification time.

EXPIRESACTIVE DIRECTIVE

The first step is to make sure that Expires headers will be generated. To ensure they are, you must activate them as in the following example:

```
<IfModule mod_expires.c>
      ExpiresActive on
</IfModule>
```

Alternatively,

```
      ExpiresActive off
```

will turn off Expires header generation.

The ExpiresActive directive can be specified anywhere within the Apache configuration hierarchy from server level down through virtual host, directory or .htaccess file. Lower level directives will override higher level directives so that, for instance, an ExpiresActive directive at the virtual host level will override the server level directive within that virtual host. Note that all the mod_expires directives operate within the Apache configuration hierarchy in this same way, with lower level directives overriding higher level directives.

Now that Expires headers are enabled, you must specify rules for generating them. This is done with the ExpiresDefault and ExpiresByType directives.

EXPIRES SYNTAX

Documents can be expired based on the amount of time elapsed from either a) the last time the client accessed the document; or b) the last time the server modified the document.

There are two different syntaxes available to specify expiration. The first uses a code ('A' or 'M') to specify one of the two types above (access or modification); followed by the number of seconds to add to the base time to calculate the expiration time. For example:

- A604800 means expire the document 1 week after access; while
- M604800 means expire the document 1 week after it is modified.

It is important to repeat that the "by access" method is determined at the client side while the "by modification" method is determined at the server side. Care should be used when specifying the "by modification" method since all client caches will expire simultaneously and once expired, will remain expired, and therefore not cached, until the next modification. The best use case for the "by modification" method is for documents that change on a regular basis such as a weekly status report or a quarterly financial report. The "by access" method is far more commonly used.

For the mathematically challenged ("How many seconds are there in month?") among us there is an alternative syntax that is somewhat easier to use. This second syntax is "<base> plus <number> <type>" and is best explained by example. Transforming the above 1 week examples into this for format we get either:

- "access plus 604800 seconds"
- "modification plus 604800 seconds"

or, more simply

- "access plus 1 week"
- "modification plus 1 week"

With the syntax in hand let's get back to our example where we'll use the more user-friendly syntax above.

EXPIRESDEFAULT DIRECTIVE

The ExpiresDefault directive is used to set the default expiration time for all documents within the realm of the directive unless otherwise overridden by an ExpiresByType directive.

```
<IfModule mod_expires.c>
 ExpiresActive on
 ExpiresDefault "access plus 4 hours"
</IfModule>
```

EXPIRESBYTYPE DIRECTIVE

Use the ExpiresByType directive to override ExpiresDefault by type of document thus gaining a finer grain control of caching behavior.

```
<IfModule mod_expires.c>
      ExpiresActive on
      ExpiresDefault "access plus 4 hours"
      ExpiresByType image/gif "access plus 1 month"
      ExpiresByType image/jpeg "access plus 1 month"
      ExpiresByType image/tiff "access plus 1 month"
      ExpiresByType image/bmp "access plus 1 month"
      ExpiresByType text/css "access plus 1 hour"
      ExpiresByType application/x-javascript "access plus 1 hour"
</IfModule>
```

Here we are expiring images 1 month after they are accessed and style sheets and javascript 1 hour after access.

MOD_HEADERS MODULE

So far we have activated Expires headers, set default expiration for all documents and overridden the default expiration for some document types. The unset, set and append options of the Header directive within the mod_headers module when used in conjunction with the FilesMatch directive, allows us to override our ExpiresByType directives to give us the finest grain cache control possible. Consider the following two use cases.

Let's assume our working example was specified at the server level, meaning it applies to all virtual hosts on the server. Let's further assume that we have one web site that handles sensitive content that we prefer not to cache on client machines. Assuming that site has its own virtual host we can specify the following directives within that site's virtual host:

```
<FilesMatch "\.(html?|gif|jpe?g|jsp?|css|js|pdf)$">
      Header unset Cache-Control
      Header append Cache-Control "private, no-cache, no-store"
      Header append Pragma "no-cache"
</FilesMatch>
```

These directives unset any previous cache control headers and append new headers indicating not to cache the selected file types.

Finally, within this same virtual host, there is a particularly large file that is frequently downloaded and you want to override the above no-cache directives just for this file. The following directives will accomplish this:

```
<FilesMatch "myLargeFile\.pdf">
      Header unset Cache-Control
      Header unset Pragma
      Header append Cache-Control "public"
</FilesMatch>
```

ISSUES WITH ORACLE SINGLE SIGN-ON (SSO) AND ORACLE CONTAINERS FOR J2EE (OC4J)

There are a couple of common issues that come up with using SSO and or OC4J together with OHS. This section covers two of the most common.

ORDER OF LOAD MODULES

The proper order of load modules is critical for the functioning of SSO and OC4J. Specifically, the SSO modules must be loaded before the OC4J module which in turn must be loaded before mod_rewrite. The following example shows the proper configuration:

```
# Include the configuration files needed for mod_osso
include "/OH/ohs/conf/mod_osso.conf"
include "/OH/ohs/conf/sso_apache.conf"

# Include the configuration files needed for mod_oc4j
# Loading mod_oc4j.conf here because it must load AFTER mod_osso.conf
include "/OH/ohs/conf/mod_oc4j.conf"
# Loading mod_rewrite module here as it has to load after mod_oc4j
LoadModule rewrite_module modules/mod_rewrite.so
```

This issue is sometimes further confused by the default configuration that some versions of OHS provide. See one of those below:

```
# Include the mod_oc4j configuration file
include "d:\infra904\Apache\Apache\conf\mod_oc4j.conf"
# Include the mod_dms configuration file
include "d:\infra904\Apache\Apache\conf\dms.conf"
# Loading rewrite_module here so it loads before mod_oc4j
```

```
LoadModule rewrite_module modules/ApacheModuleRewrite.dll
```

Note the confusing last comment above. It says that rewrite must be BEFORE OC4J but it actually puts it after OC4J. The comment is not correct.

SSO, Pragma "no-cache", and Misbehaving Browsers

SSO by default will always prevent client side caching of SSO-protected content. This is a good security practice since it prevents information fragments from being cached on local machines. However, with some versions of Internet Explorer (IE), this behavior can cause problems.

If you use SSO and IE and your application allows the download of Word, Adobe or other documents that IE reads via plug-in, users will sometimes get the following error when attempting to open the document:

```
"Internet Explorer cannot download xxxx.pdf from yoursite.com
Internet explorer was not able to open this internet site. The requested site
is either unavailable or cannot be found. Please try again later."
```

The only workaround for this issue is to allow the file to be cached locally. There are two ways to accomplish this. If you are using SSO, you can use the

```
OssoSendCacheHeaders off
```

directive. This directive will override the default SSO behavior and SSO will stop sending Pragma "no-cache" headers for SSO-protected content. For security reasons, be judicious in the use of this directive. It should be used in the narrowest context as possible by embedding it in other Apache directives.

For example, if the above file is the only file downloadable, use a FilesMatch directive as follows:

```
<FilesMatch "xxx\.pdf">
      OssoSendCacheHeaders off
</FilesMatch>
```

If you have multiple Word and Adobe files then do it by file type as in:

```
<FilesMatch "\.(doc|pdf)$">
      OssoSendCacheHeaders off
</FilesMatch>
```

Note this poor browser behavior is possible whenever you allow document downloads with "no-cache" headers which you may chose to do even if you are are not using SSO. The above solution is SSO specific. If you encounter this same issue and you are not using SSO, simply use the unset and append directives to fix the problem as in:

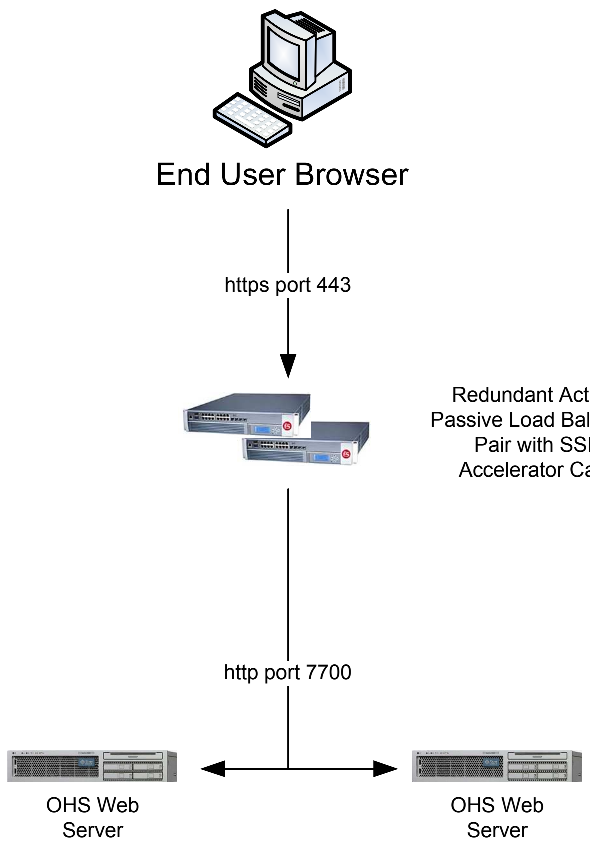
```
<FilesMatch "xxx\.pdf">
      Header unset Cache-Control
      Header unset Pragma
      Header append Cache-Control "public"
</FilesMatch>
```

or

```
<FilesMatch "\.(doc|pdf)$">
      Header unset Cache-Control
      Header unset Pragma
      Header append Cache-Control "public"
</FilesMatch>
```

Terminating SSL in Front of OHS

A typical production environment will have multiple servers running OHS to provide scalability and high availability. In such environments a hardware load balancer is often used to both monitor the health of and distribute traffic to the pool of web servers. Because encryption via hardware SSL accelerator cards is much faster than software-based SSL encryption, the load balancer is often used as the SSL termination point. In this case all traffic between the load balancer and the client browser is encrypted (https) but from the load balancer to OHS the traffic is in the clear (http) as in the diagram below:



In this configuration, OHS is not aware of the SSL terminating above it and will therefore send any self referencing URLs via http instead of https thus causing problems at the browser. To make OHS aware of the SSL in front of it, use the SimulateHttps directive as follows:

```
<VirtualHost MyVirtualHost:7700>
      SimulateHttps on
      .
      .
</VirtualHost>
```

This directive can also be implemented at lower levels using `Location` or `Directory` directives and will require loading of mod_certheaders.

QUARTER

PHASE 4: PLANNING FOR THE FUTURE

CHANGES WITH 11G BASED ON APACHE 2.2.13

Taking a tour of the latest Oracle HTTP Server (OHS) 11.1.1.2 release from a security perspective. This uses a simple red, yellow, green scale to assess how that configuration item was addressed.

It has been a long time since a new version of OHS has come out. The previous OHS versions based on Apache 1.3 and 2.0 were a little tired. The web server is your front door. Let's see how the security of the new OHS version based on Apache 2.2.13 stacks up. This first look was based on installing the web tier configuration on Linux.

Configuration Item	Rating
Apache Version - The latest release is based on Apache 2.2.13, which is a very recent version. This includes a fix for a vulnerability in mod_status that Apache fixed in 2.2.11.	🟢
Where is mod_security? -The doc says version 2.5.9 is included in the distribution. However, if you check you will see the module is not there. The previous versions of OHS distributed an old version of mod_security (1.84), which was still very usable. We used mod_security to setup filtering rules including a rule that stopped an XSS vulnerability in OBIEE. We opened an SR and Oracle replied that mod_security would not be distributed with FWM 11g. Integrating the open source version is not supported, so if you were using this module you are out of luck.	🔴
Conf file macros - The httpd.conf file uses macros to set the Oracle Home directory for directives such as the DocumentRoot. This means other tools cannot understand the format. We always used the .apachectl configtest option to verify the config before restarting OHS. This no longer works with the macro format. The macro option is not a bad choice, but why not go with mod_macro?	🟡
Compile options - Oracle compiled a minimum number of modules in the distribution. It's basically just the core modules and two Oracle specific modules, odl_log and ora_audit. You can see the list with the command httpd.worker -l. This gives you the option to disable modules that are not needed for the intended use. Oracle also compiled with the threaded MPM worker by default, unlike the previous OHS 2.0 version that required OPMN changes to enable threads.	🟢
Wildcard include - The httpd.conf uses a wildcard to include conf files from the moduleconf sub-directory. This is really a bad idea. If someone inserts a bad conf file into the moduleconf sub-directory then the next time the OHS server restarts it will pick up this file and use it. The configuration files that are used should be explicitly named.	🔴
Server Signature - The config directives were set more securely with the addition of ServerTokens Prod and ServerSignature off.	🟢
cgi-bin - The printenv and test-cgi scripts are still being distributed in the cgi-bin directory. In addition, the cgi-bin directory is uncommented in the httpd.conf file. The only thing preventing execution of these scripts is setting execute permission on the file. These files should not be included in the distribution.	🟡
Logging - The default logging is ODL style with an option to switch to standard Apache logging. Using ODL logging can limit the options for using third party analysis tools. This just does not look like it adds real value. The XML style logs just seem to add more overhead.	🟡

OHS and WebLogic - mod_wl module

The mod_wl module provides a connector between OHS and WebLogic similar to the functionality provided by mod_oc4j. Oracle is making the transition from OC4J to WebLogic. The mod_oc4j module is not being distributed with the latest version of 11gAS. If you are transitioning from OC4J it is still possible to run a version of OHS that supports both OC4J and WebLogic following MetaLink note #796072.1. This uses a version of OHS based on Apache 2.0. The 11gAS version distributes an Apache version based on 2.2.13. Apache modified the module API between 2.0 and 2.2. This means mod_oc4j will not run with the latest distribution.

There are two versions of the mod_wl module. The older version is called mod_wl, while the newer version distributed with 11gAS is mod_wl_ohs. The older version had a number of significant vulnerabilities reported against it. If you need to run the older mod_wl module, then make sure you are patched current to pick up the latest fixes. The mod_wl_ohs module should be considered a different module since it includes features that did not exist in the earlier product. Another thing to consider if you will be running the older mod_wl module, there is one version of the module for Apache 2.0 and another for Apache 2.2 because of the API change. Oracle also distributes a version that is labeled "regular strength encryption" and another with 128-bit encryption. Therefore if you are looking to use a version with an OHS based on Apache 2.2, and you need 128-bit encryption you should be looking for the module mod_wl128_22.so.

The Fusion Middleware release notes for 11g highlight a few of the considerations. As an example, the failover operates at a container level, and not an application level. If you stop an application, requests will continue to be routed to the container as long as it is up, and end users will likely receive 404 errors. Another note indicates that the WebLogic configuration is configured by default in httpd.conf. If you do not plan to use this, you will need to modify the configuration to disable it. One thing to watch out for is module naming. Many references in the documentation identify the new version of the module as mod_wls_ohs, but if you look at the actual web tier distribution the module is called mod_wl_ohs. If you are looking for information on this module you may want to search under both names.

The mod_wl_ohs module includes a number of configuration parameters. These are documented in the General Parameters for Web Service Plug_ins. There is a large list, but we will mention a few that relate to security and that you should review when setting up your site. When using mod_oc4j you could check routing information using the static mod_oc4j.conf file or using the command opmnctl debug. With WebLogic there is a configuration parameter, DebugConfigInfo, that enables this debug information. This Metalink note describes how to set it: How to View FMW 11g Web Tier Mod_WebLogic Routing and Other Debug Information? [ID 967889.1]. This is the type of directive that you may want to set conditionally with an IfDefine, so it does not inadvertently get turned on in the production environment. The parameters include definition of an ErrorPage, which will be displayed when the web server cannot forward the request to WebLogic. Remember one of the guidelines is to replace default pages with custom error pages to prevent information leaks. There is a routing parameter, WLLocalIP, that controls which network path is used on a multi-homed node. Without this setting a random selection will be made, which is probably not what you want. There is another feature of mod_wl_ohs that will cache large POST data requests, those over 2k, in the TEMP directory to support failover. This directory is world readable, so you should explicitly set your own location for TEMP rather than take the default. On the surface this feature seems vulnerable to attack because it allows a web site user to upload a file onto the web server into a known location. This file could contain data that has the potential to compromise the web site.

As you can see there are significant changes with the new mod_wl_ohs module that you need to consider when beginning an implementation. Since this is a brand new module we anticipate patches will need to be issued. We would encourage you to frequently check Metalink to stay up-to-date on the latest information about this module.

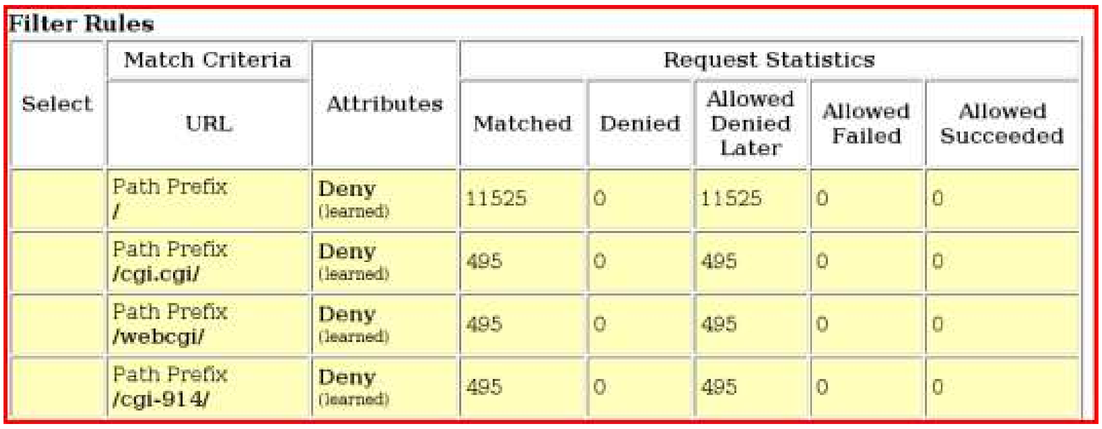
Web Cache 11g URL filtering feature

Did you ever see the movie WarGames? This featured a computer called the WOPR. This was a good movie. There was a nice balance between the human story and the computer. Most of the movie Twister was good, up until the end. Both movies climaxed with flashing computer lights. In Twister it was just numbers scrolling on a laptop, which was lame. In WarGames the flashing lights showed simulated attacks demonstrating the computer learning, and worked well.

Two recent things brought back memories of the movie WarGames. One item was an article in the Washington Post about students hacking into the school computer and changing grades. The other was the new release of Oracle WebCache that includes a learning rule capability for request filtering. In the movie, they use Tic-Tac-Toe to help the computer learn. To try out the new WebCache learning mode we chose Nikto. This is a web security scanner that sends thousands of requests to test the security of a web server. This tool is easy to setup and use, and it rhymes with Tic-Tac-Toe. Let's try it against the Webcache Oracle Process for Request Filtering (WOPRF), and see how it does.

We fired up the new WebCache 11g and chose the new Request Filters menu option. We looked to see what rules were provided out of the box. There are less than 10 rules defined! Checked Metalink for any additional information about the learned rules feature and did not find anything. Reading through the documentation it does not explain how this feature works. Looking at the rules that were defined it included one for the Trace method and another for checking for null bytes. Those are keepers. If you add a rule and then check the configuration to see the rule format, the new rule is added to the webcache.xml file. It's too bad the rules were not split out into a separate XML file. Most of the WebCache configuration does not change, but the rules are likely to be updated more regularly. Managing these in an independent file would have been much better. If you add rules to block bots and spiders, and other rules for cross site scripting and SQL injection then this will add a lot of entries to the file.

There are two categories for learned rules, method and URL. We ran Nikto and it sent thousands of requests. After the test was completed we reviewed the learned rules. The Method category showed learned rules for all the HTTP methods (OPTIONS, PUT, etc.) with a suggested action of Allow. The URL category showed 25 learned rules. This included a number of useful Deny rules for prefixes like cgi-bin and scripts. The following screen shot shows the URL category with some of the learned rules.



Filter Rules

Select	Match Criteria	Attributes	Request Statistics				
	URL		Matched	Denied	Allowed Denied Later	Allowed Failed	Allowed Succeeded
	Path Prefix /	Deny (learned)	11525	0	11525	0	0
	Path Prefix /cgi.cgi/	Deny (learned)	495	0	495	0	0
	Path Prefix /webcgi/	Deny (learned)	495	0	495	0	0
	Path Prefix /cgi-914/	Deny (learned)	495	0	495	0	0

Figure 1 - WebCache 11G URL Filtering Learned Rules

Overall WOPRF was more like Twister rather than WarGames. If you use mod_security it's easy to find a large number of rules that can be plugged right in and used. You would either have to create these rules in WebCache or convert the mod_security rules to WebCache format. Another limitation is the filtering is limited to requests. This means the response cannot be filtered. Filtering responses to prevent information leaks with error messages is a basic feature that needs to be there.

If you have a strong filtering mechanism in place to protect the web site, then you can achieve your goal of reaching a draw between people trying to attack the web site and your defenses. This may convince them that the only winning move is not to play.

REFERENCES

ORACLE METALINK REFERENCE NOTES

ORACLE HTTP SERVER (OHS)

Doc ID: 260449.1 - Subject: Everything You Wanted to Know About the Apache-Based OHS Version

Doc ID: 334763.1 - Subject: How Apache Works

Doc ID: 258231.1 - Subject: How To Change The Information That Apache Returns About Itself In The Header ?

Doc ID: 314381.1 - Subject: How to Setup Oracle HTTP Server as a Virtual Host Reverse Proxy

Doc ID: 378003.1 - Subject: How to Configure Virtual Hosts on HTTP_Server for LBR Access or SSL Termination

Doc ID: 428391.1 - Subject: Is it Possible To Configure Mod_plsql With Apache 2.0?

Doc ID: 293697.1 - Subject: Preparing and Configuring Virtual Hosts on Oracle Application Server 10g

Doc ID: 400010.1 - Subject: Steps to Maintain Oracle Database 10.2 Companion CD Home

MOD_SECURITY

Doc ID: 456388.1 - Subject: How to enable and test mod_security in HTTP Server

Doc ID: 733329.1 - Subject: How To Enable Logging When mod_security is Enabled?

Doc ID: 389558.1 - Subject: Cannot Access Pls Pages: 'mod_security: Access denied with code 400'

MOD_REWRITE

Doc ID: 259404.1 - Subject: How to Redirect Based on REQUEST_METHOD: Disabling HTTP TRACE

Doc ID: 460564.1 - Subject: Hints and Tips for Troubleshooting the URL Firewall

Doc ID: 310969.1 - Subject: How To Configure & Run Apache in Restricted Mode During Maintenance

E-BUSINESS SUITE

Doc ID: 239176.1 - Subject: Dual OHS Listener Configuration for Mid-Tier PL/SQL in E-Business Suite 11i

Doc ID: 726711.1 - Subject: Mod_plsql and Oracle E-Business Suite Release 12

Doc ID: 287176.1 - Subject: DMZ Configuration with Oracle E-Business Suite 11i

ADDITIONAL REFERENCE LINKS

1. Apache 2.0 Documentation - Multi-Processing Modules
2. A Complete Guide to the Common Vulnerability Scoring System Version 2.0, CVSS Calculator
3. Recommended Security Controls for Federal Information Systems - NIST 800-53
4. Center for Internet Security
5. Apache Module Mod_rewrite
6. Mod_rewrite URL Rewriting Guide
7. Mod_security
8. Got Root Site maintaining a set of Mod_security rules
9. Apache Security Book by Ivan Ristic
10. Preventing Web Attacks with Apache by Ryan Barnett
11. Marcus Ranum on computer security

TOOLS

CHECKING CONFIGURATION

HTTPrint - http://www.net-square.com/httprint/

Nikto Web Security Scanner

Additional information and scripts can be found on Secure DBA

LOG ANALYSIS

WebLog Expert

find_status.pl Perl script and other tips

Log Parser from Microsoft

ABOUT THE AUTHORS

KEVIN SHEEHAN

Kevin has over 29 years experience in Information Technology. That makes him either a grizzled veteran or an old coot depending on your state of mind. His years at AT&T, Oracle, Unisys and Agilex Technologies has given him experience on platforms ranging from IBM mainframes to Unix-based mid-range servers to Unix-based mainframes to Linux-powered grid technologies.

Kevin has worked with Oracle technology for the past 16 years and for the last seven years has worked in the federal government space, particularly, Homeland Security environments. He is familiar with securing systems based on the Department of Homeland Security's Secure Baselines as well as the controls identified in NIST 800-53 - Recommended Security Controls for Federal Information Systems and Organizations and takes a special interest in application infrastructure and service and performance monitoring.

Kevin blogs; however irregularly, at http://securedba.com and can be reached at kevin.sheehan@agilex.com. His LinkedIn profile can be found at http://www.linkedin.com/in/kevinpsheehan. Agilex Technologies can be found on the web at http://agilex.com.

BRIAN J. MULREANY

Brian started his career with AT&T Business Services Division. He held a number of positions with AT&T including Technical Director for Software Architecture. During his career at AT&T he managed systems to support the 800 Inbound Calling service. Brian left AT&T and moved to Oracle Consulting right when the Internet took off. Those were exciting times with new companies and new ideas being launched every day. At Oracle, he worked in the commercial sector with a wide variety of companies and industries deploying Internet facing applications based on Oracle and Java. Brian joined Unisys as a system architect right after Unisys won the TSA contract. He currently holds a senior position at Unisys supporting the TSA contract.

Email Contact: bjm-uva@alumni.virginia.edu

LinkedIn Profile: http://www.linkedin.com/in/brianjmulreany

Follow Brian in the blogosphere: http://bjm-battle-room.blogspot.com/